2-2015

Preconditioning For Matrix Computation

Xiaodong Yan
Graduate Center, City University of New York

PRECONDITIONING FOR MATRIX COMPUTATION

by

XIAODONG YAN

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

2015

This manuscript has been read and accepted for the
Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

Victor Pan

January 29, 2015

Date | Chair of Examining Committee

Robert Haralick

Date | Executive Officer

Theodore Brown

Delaram Kahrobaei

Florian Lengyel

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

PRECONDITIONING FOR MATRIX COMPUTATION

by

Xiaodong Yan

Advisor: Professor Victor Pan

Preconditioning is a classical subject of numerical solution of linear systems of equations. The goal is to turn a linear system into another one which is easier to solve.

The two central subjects of numerical matrix computations are LIN-SOLVE, that is, the solution of linear systems of equations and EIGEN-SOLVE, that is, the approximation of the eigenvalues and eigenvectors of a matrix. We focus on the former subject of LIN-SOLVE and show an application to EIGEN-SOLVE. We achieve our goal by applying randomized additive and multiplicative preconditioning.

We facilitate the numerical solution by decreasing the condition of the coefficient matrix of the linear system, which enables reliable numerical solution of LIN-SOLVE.

After the introduction in the Chapter 1 we recall the definitions and auxiliary results in Chapter 2. Then in Chapter 3 we precondition linear systems of equations solved at every iteration of the Inverse Power Method applied to EIGEN-SOLVE. These systems are ill conditioned, that is, have large condition numbers, and we decrease them by applying randomized additive preconditioning. This is our first subject.

Our second subject is randomized multiplicative preconditioning for LIN-SOLVE. In this way we support application of GENP, that is, Gaussian elimination with no pivoting, and block Gaussian elimination. We prove that the

proposed preconditioning methods are efficient when we apply Gaussian random matrices as preconditioners. We confirm these results with our extensive numerical tests. The tests also show that the same methods work as efficiently on the average when we use random structured, in particular circulant, preconditioners instead, but we show both formally and experimentally that these preconditioners fail in the case of LIN-SOLVE for the unitary matrix of discreet Fourier transform, for which Gaussian preconditioners work efficiently.

Acknowledgments

I would like to thank my advisor, Dr. Victor Pan for his guidance through this work. Without his encouragements and support this dissertation would not have been possible. I also want to thank my committee members, Dr. Theodore Brown, Dr. Florian Lengyel, and Dr. Delaram Kahrobaei for their suggestions and feedback.

Last but not least, I would like to thank my wife Dr. Yi Yan. Thank you for putting up with me and taking good care of our young daughters when I was absent from my fair share of parental duties working on this dissertation.

To my kids Katie & Amber - Yes, daddy has more time to play with you now.

To my parents - This is for you also!

Contents

List of Figures

List of Tables

1 Introduction

1.1 Overview

Matrix computation has frequently turned out to be the most important part of computation for complex science and engineering problems. The recent hot topics of "searching algorithms", "big data", "machine learning" also use matrix computation extensively as the key steps for finding solutions, which often involve eigen-solving, singular-value decompositions and other matrix computations. New developments in hardware architecture and dramatic increase of the data size of the real world problems have also lead to research on more efficient methods for matrix computation. With this development the readily available "efficient" default methods, even those in commercial software packages, are no longer satisfactory. Preconditioning technique is explored in order to improve the known algorithms for matrix computations. The first part covers our advance in additive preconditioning, which has been the topic in the early stage of the author's research effort. A lot of progress has been made on using (random) additive preconditioner for eigen-solving, null space solving and matrix approximation. The inverse power iteration method easily leads to ill-conditioned linear system solving. We combine the additive preconditioning methods with this method to arrive at well conditioned linear systems solving and tested our results numerically. We advanced in this research area and published several papers, jointly with our advisor.

The most recent research effort of the author has been the application of random multipliers to some central problems of matrix computations.

1.2 Preconditioning Method, linear system solving and Eigen-solving with Inverse Power Iteration

Preconditioning is a classic subject of numerical solution for linear systems of equations $Ax = b$. One modifies the input matrix A in order to decrease its condition number. Better conditioned linear systems, that is, ones having smaller condition numbers, can be solved more accurately and faster. Traditional preconditioning is the transition to a simpler linear systems $MANx = Mb$ such that $y = Nx$. The critical problem for preconditioning is the choice of the multipliers M and N above (one of them can be the identity matrix) that would decrease the large condition number $cond(A)$ to a much smaller value $cond(MAN)$ or would compress the spectrum of the singular values of the matrix A into a small number of clusters. The early choices usually involved factorization or inversion of the matrix A, which is generally as expensive as the solution of a linear system, can be unstable numerically, and may not always be an ideal choice for sparse matrix or matrix with structures that can be taken advantage of. As an alternative or complementary tool, we explored *random additive pre-processing* $A \leftarrow C = A + P$, i.e., we add a matrix P (*additive preprocessor* with a smaller rank or being a structured matrix) to the input matrix A to obtain a new matrix C with a smaller condition number. We explored different additive conditioners in our research, and studied them theoretically, and numerically tested their efficiency. Null space solving is also studied and an approach using additive preconditioning has been proposed and supported with a formal proof. We applied these results to support the popular Inverse Power Method for approximation to the eigenvalues and eigenvectors of a matrix, which involves the solution of ill conditioned linear system of equations. Those formed the first part of the thesis.

1.3 Multiplicative Preconditioning for low-rank approximation and GENP

Linear system solving is a fundamental topics for many branches of science and engineering. Direct solution using classic Gaussian Elimination Method has been one of the most popular method throughout the centuries [TB97]. However it can easily run into many numerical stability problems unless one applies pivoting, that is, row and column interchange of the input matrix. This occurs frequently even for some small dimensional problems. Also the computational cost for large problems has become mostly dependent on data access layer rather than flops time since the data tend to be so large that accessing the hard drive becomes the bottleneck of the computations. Presented here are the results using random multiplier in order to avoid these problems even for the Gaussian Elimination with No Pivoting (GENP). Formal and empirical support is shown for the application of the Gaussian random multipliers, and empirical support for the application of random structured multipliers, which use fewer random parameters.

We also explored random multipliers as a tool for low rank approximations of a matrix having a small numerical rank. Randomized algorithms for this task have been studied extensively, both formally and experimentally [HMT11]. They are numerically stable, run at a low computational cost, and allow low-cost improvement of the output accuracy by means of the Power Method. This can be used in many important applications such as data mining, statistics, PDEs and integral equations. By extending the analysis of preprocessed GENP, we prove that even without the customary oversampling the algorithms are expected to output the desired low-rank approximations to the input matrix in our tests. Both Gaussian and Toeplitz random multipliers are used and Toeplitz random multipliers have been shown as effective based on the numerical

test results.

1.4 Organization of the thesis

In Chapter 2 the basic definition and terminology used are defined. Some background on matrix computation are discussed. In particular conditioning of a matrix and a linear system is discussed. In Chapter 3 the results on additive preconditioning are presented. Most of the research outcome in Chapter 3 has been published in the papers [PKMRTCY], [PIMRTY], [PIMRTTY], [PY07], [PY09], [PIM10]. In Chapter 4 we discuss using random multiplier with GENP, low-rank approximate of matrix. Tables and figures for methods discussed for GENP and low-rank approximation are included in this thesis. Most of the material in Chapter 4 has been presented in the paper [PQYa] (submitted).

2 Definitions

2.1 Basic Definitions

Our basic definitions reproduce or slightly modify the customary definitions in [GL96], [TB97], [W02] for matrix computations. We use the same definition for Hermitian, unitary (orthogonal), singular, full-rank and rank deficient matrices. I_k is the $n \times n$ matrix. O_k, l si the $k \times l$ matrix filled with zeros. We write I and O if the matrix size is defined by context. $A_{k,l}$ denotes its leading, that is, northwestern $k \times l$ block sub-matrix, and we also write $A^{(k)} = A_{k,k}$.

A^T is the transpose of matrix A (Hermitian transpose). $||A|| = ||A||_2$ is the spectral norm of a matrix A. $\rho = \operatorname{rank} A$ is the rank of the matrix. Singular values $\sigma_j(A)$, $j = 1, \dots, \rho$, in non-increasing order for matrix A where 2-norm $||A|| = \sigma_1(A)$. $||A||_F$ is its Frobenius norm and defined as $||A||_F = \sqrt{\sum_{i=i}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$.

We use $A^+ = (A^H A)^{-1} A^H$ for the *Moore-Penrose generalized inverse* of matrix A (also called *pseudo inverse*).

$A^+ = (A^H A)^{-1} A^H$ if $m \geq n = \rho$, $A^+ = A^H (A A^H)^{-1}$ if $m = \rho \leq n$, and $A^+ = A^{-1}$ if $m = n = \rho$.

A matrix A is normalized if $||A|| = 1$. We write $M \geq 0$ for a non-negative definite Hermitian matrix M and we write $n \gg d$ where the ratio n/d is large. $\operatorname{diag}(B_1, \dots, B_k)$ and $\operatorname{diag}(B_i)_{i=1}^k$ denote the $1 \times k$ block matrix with the blocks $B_1, \dots, B_k$ and $k \times k$ block diagonal matrix with the diagonal blocks $B_1, \dots, B_k$, respectively. We write (B, C) to denote the 1×2 block matrix with the blocks B and C. A matrix A is a matrix basis for its range if its columns are linearly independent. A *null matrix basis* for a matrix is a vector, a basis, and a matrix basis for its (right) null space, respectively. Similar concepts are defined for the left null space. We write $Q(M)$ for the Q-factor of the size $m \times n$ in the thin QR factorization of an $m \times n$ matrix M of the full rank where

the R-factor has positive diagonal entries. We use the standard term of full SVD and compact SVD for singular value decomposition.

2.2 Conditioning of a Linear System and of a Matrix

Generally in numerical computing, the "condition of the problem" measures how accurately one can solve the problem with the floating point precision regardless the algorithms used. For a function

$y = f(x)$, we have $(y + \delta y) = f(x + \delta x)$,

Denote the machine's epsilon as ε then δx satisfies $\frac{|\delta x|}{|x|} \leq \varepsilon$.

The condition number $\kappa_f(x)$ of the problem at x is the ratio of $\frac{|\delta y|}{|y|}$ to $\frac{|\delta x|}{|x|}$. It measures approximately how much the relative rounding error in x is magnified by evaluation of f at x[O01].

For linear system solving with a square matrix A we have the equations $Ax = b$ and $A(x + \delta x) = b + \delta b$, those two equations imply that $A\delta x = \delta b$, so $\delta x = A^{-1}\delta b$. From this we can have

$||\delta x|| \leq ||A^{-1}||\ ||\delta b||$.

Also from $||b|| = ||Ax|| \leq ||A||\ ||x||$ we can have

$\frac{1}{||x||} \leq ||A||\frac{1}{b}$.

Multiply the two equations above and arrive at

$\frac{||\delta x||}{||x||} \leq ||A||\ ||A^{-1}||\frac{||\delta b||}{||b||}$.

where $\kappa(A) = ||A||\ ||A^{-1}||$ is the condition number of the matrix A.

If $\kappa(A)$ is large, then the matrix problem is called *ill-conditioned*, otherwise *well-conditioned*. Here "large" is understood in context.

A lot of algorithms have been developed in order to estimate the condition number of a matrix *quickly* without computing the norms for both A and A^{-1}, such as the "RCOND" procedure in BLAS/LAPACK and MATLAB (it estimates the reciprocal

of condition number of a matrix.)

We call left nullity as $\operatorname{lnul} A = m - \rho$, right nullity $\operatorname{rnul} A = n - \rho$, and nullity $\operatorname{nul} A = \min\{m, n\} - \rho$. We say that $r = \operatorname{nnul} A$ is the numerical nullity and $l - r = \operatorname{nrank} A$ is the numerical rank of the matrix A if the ratio $\sigma_1(A)/\sigma_{l-r}(A)$ is not large, whereas $\sigma_1(A) \gg \sigma_{l-r+1}(A)$, that is, if the matrix has exactly r singular values that are small relative to $||A|| = \sigma_1(A)$, say are less than $\tau||A||$ for a fixed small positive tolerance value τ.

3 Eigenspaces, Inverse Iteration, Additive Preconditioning and Auxiliary Results

This section discuss the research done using additive preconditioning method. It start with an overview of additive preconditioning idea followed by the summary of the popular Inverse Power Iteration. Numerical results are presented for additive preconditioning approach. Lastly the null space solving using Additive Preconditioning method are discussed with some theorems and algorithms.

3.1 Additive Preconditioning of linear systems of equations and extension to eigen-solving

As mentioned in introduction part, a matrix $C = A + P$ is a preconditioned matrix for matrix A of size $n \times n$. We also call P additive preprocessor (APP). For a positive integer r we define two generator U and V of the size $n \times r$, let $P = UV^H$ so a new modified matrix is $C = A + UV^H$. According to our analysis and experiments [PIMRTY], adding a random and properly scaled matrix UV^H of a rank r (such that $\frac{||UV^H||_2}{||A||_2}$ is neither large nor small) is expected to decrease the condition number $cond(A)$ $=\frac{\sigma_1(A)}{\sigma_n(A)}$ to the order of $\frac{\sigma_1(A)}{\sigma_{n-r}(A)}$ where $\sigma_j(M)$ denotes the jth largest singular value of a matrix M. If $\sigma_{n-r}(A) \gg \sigma_n(A)$ then the condition number is decreased substantially.

We introduce this A-preconditioning (that is, additive preconditioning) in order to accelerate the inverse power iteration for eigen-solving. For an $n \times n$ input matrix M every iteration step essentially amounts to the solution of a linear system of equations with the matrix $A(\tilde{\lambda}) = \tilde{\lambda} I_n - M$, whose conditioning rapidly deteriorates, that is, whose condition number rapidly increases, as the approximation $\tilde{\lambda}$ converges to an eigenvalue λ. Solving such a linear system is a hurdle, even though the scaled solutions

rapidly converge to an eigenvector in spite of the rounding errors. In our modification we yield the same convergence rate, but solve well conditioned linear system with the coefficient matrices $C(\tilde{\lambda}) = A(\tilde{\lambda}) + UV^H$. Two approach are proposed, and one of them is implemented by the author. Approach 1 uses the Sherman-Morrison-Woodbury inversion formula

$A^{-1} = (C - UV^H)^{-1} = C^{-1} + C^{-1}UG^{-1}V^HC^{-1}$,

with $G = I_r - V^HC^{-1}U$.

Approach 2 approximates the eigenvectors associated with an eigenvalue λ of matrix M by the solutions of linear systems $Cy = u$ where $u = Ux$ for some vectors x. This new approach covers the case of simple, multiple, and clustered eigenvalues. As a natural extension, the null space computations are also discussed at the end.

3.2 The Inverse Power Method for Eigen-solving

A lot of efforts have been spent by researchers to devise efficient and reliable eigen-solving algorithms. The classical (now old-fashioned) approach is to begin with computing the characteristic polynomial, but this stage is numerically unstable and is by-passed in the present day approaches [W02],[P80]. Three widely used methods during the past several decades are recalled next.

3.2.1 The Power method and Google's pagerank

The Power Method or Power Iteration is the classical iterative method for computing the eigenpair for the absolutely largest eigenvalue of a matrix. It is also involved in devising some most used eigen-solving algorithms such as the Inverse power iteration and QR-methods [TB97]. The Power iteration takes the following steps:

Algorithm 3.1. Power iteration

$\nu \longleftarrow$ *a scaled random vector satisfying* $||\nu||_2 = 1$

for $k = 1, 2, \ldots$

1. $\omega_k = A\nu_{k-1}$
2. $\nu_k = \omega_k / ||\omega_k||$
3. $\mu_k = (\nu_k)^T A \nu_k$

until $||A\nu_k - \mu_k \nu_k|| < \tau ||A|| \cdot ||\nu_k||$, *where* τ *is a fixed positive tolerance.*

Convergence analysis: Let us denote the eigenvalues as λ_i, ordering them according to the absolute values such that

$|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$.

The corresponding eigenvectors are u_i. A random vector ν_0 can be decomposed as

$\nu_0 = \sum \alpha_i u_i$.

After repeated multiplication of the matrix A k times we have

$$\nu_k = \beta_k \cdot \lambda_1^k [\alpha_1 u_1 + \sum_2^n \alpha_i (\lambda_i / \lambda_1)^k u_i],$$

where $\beta_k = \prod ||\omega_k||^{-1}$ is the multiplier, such that for all i we have $||\nu_i|| = 1$. If k is large enough as $(|\lambda_i / \lambda_1|)^k \to 0$, then ν_k and μ_k will be a good approximation of the eigenpair (λ_1, ν_1) of A, associated with the absolutely largest eigenvalue λ_1. The convergence rate is linear, proportional to the ratio $|\frac{\lambda_1}{\lambda_2}|$ and the convergence is slow when this ratio is close to 1. Because of this reason other methods such as Rayleigh Quotient Iteration have been proposed in order to achieve faster convergence and to approximate other eigenvalues besides λ_1 directly, without deflation.

It is noted that in the last few years the Power Iteration gained new interest of some researches because it was used in the PageRank algorithms for web search engine such as Google [PB99], [KHG04].

Google matrix is frequently defined as

$G = \alpha S + (1 - \alpha) u e^T$,

where $0 < \alpha < 1$, and u is an n-dimensional positive vector normalized by $||u^T||_1 = 1$. The matrix S above is an $n \times n$ column-stochastic matrix, which means that it satisfies $e^T S = e^T$ with $e^T = (1, 1, \ldots, 1)$. The eigenvector of G corresponding to the largest eigenvalue 1 is called the PageRank. The size of the matrices back to years 2000-2002 was already in the range of 27 billions, which makes the other methods unlikely to be practical. The ease of matrix-by-vector computation makes the Power Method the natural choice to apply on those sparse matrices of extremely large size.

3.2.2 The Inverse Power Method

The inverse Power iteration - IPI -is similar to power iteration but it is capable of converging to any eigenvalue by starting with an approximation to it in the desired region. If the initial approximation to an eigenvalue is good, then the IPI converges to this eigenvalue and associated eigenvector very rapidly.

"Inverse iteration is the method of choice when one has a good approximations to a subset of eigenvalues of A, and the goal is to find the corresponding eigen-vectors." The method is frequently used in structural mechanics where the extreme eigenvalues and eigenvectors are sought for a positive-definite eigen-system [TB97].

Algorithm 3.2. Inverse Power iteration (IPI).

INPUT: *A matrix A, an initial guess of μ closet to an eigenvalue λ, and a positive value τ, an positive integer γ.*

OUTPUT: *either FAILURE or an approximation $\mathbf{y}_{final}$ as the eigen-vector corresponding to λ*

INITIALIZATION: $\mathbf{y_0} \longleftarrow$ *a random vector satisfying* $||\mathbf{y_0}||_2 = 1$, $k \longleftarrow 0$.

COMPUTATIONS:

1. $k = k + 1$, *Solve* $(A - \mu I)\omega = y_{k-1}$ *for* ω

2. $\mathbf{y_k} \longleftarrow \frac{\omega}{||\omega||}$

3. *produce FAILURE if* $k > \gamma$,

   or Repeat until $||\mathbf{y_k} - \mathbf{y_{k-1}}|| \leq \tau$, *then output* $\mathbf{y}_{final} = \mathbf{y_k}$.

The classical inverse power iteration for a given approximation $\tilde{\lambda}$ to an eigenvalue λ computes an eigenvector close to the eigenvector corresponding to λ.

3.2.3 Rayleigh Quotient Iteration Method

A modification based on IPI - Rayleigh Quotient iteration (RQI) approximates both eigenvalue and eigenvector at the same time. At each step, a new approximation to an eigenvalue λ is updated with the Rayleigh Quotient based on the new approximation of the eigenvector. Due to this update, the method converges super-linearly, namely, quadratically or cubically. Several other iterative methods based on IPI also use the Rayleigh quotient but update the approximation to the eigenvalue or eigenvector slightly differently [M00]. Here the Rayleigh Quotient classical iteration:

Algorithm 3.3. Rayleigh Quotient Iteration (RQI).

INPUT: *A matrix A, an initial approximation* $\tilde{\lambda}$ *to an eigenvalue* λ, *a positive value* τ, *a positive integer* γ.

OUTPUT: *either FAILURE or an approximation* $(\lambda_{final}, y_{final})$ *to an eigen pair of* $A(\lambda)$ *such that* $||A(\lambda_{final})y_{final}||_2 \leq \tau||A||$.

INITIALIZATION: $y \longleftarrow$ *a random vector satisfying* $||y||_2 = 1$, $k \longleftarrow 0$.

COMPUTATIONS:

1. $\nu \longleftarrow (\tilde{\lambda}I - A)^{-1}y$,

2. $y \longleftarrow \frac{\nu}{||\nu||}$, $\tilde{\lambda} \longleftarrow \nu^T A\nu$, $COUNT{+}{+}$,

3. *produce FAILURE if* $k > \gamma$,

Repeat until $||A\nu - \tilde{\lambda}\nu|| \leq t||A||$, *then output* $\lambda_{final} = \tilde{\lambda}$, $y_{final} = y$.

This iteration method has been thoughtfully investigated and extensively used in computational practice. The computation of the vector ν at *step* (1) involves the solution of ill-conditioned linear system whose condition number grows to inf as $\tilde{\nu} \to \nu$. This can be computationally expensive. Furthermore rounding errors can lead to numerical problems.

For IPI the initial guess for ν may not be close enough. However having a new guess too close with PQI type of update can easily lead to ill-conditioned situation. Direct methods based on the factorization of the matrix A into easily invertible matrices are widely used and produce reliable results in predictable amount of time and storage. For a large scale and sparse problems the direct methods generally are not applicable, and iterative methods are explored. Preconditioning is widely used in the iterative methods. "It is widely recognized that preconditioning is the most critical ingredient in the development of efficient solvers for challenging problems in scientific computation, and that the importance of preconditioning is destined to increase even further" [B02], [TB97]. In next section we will study the impact of preconditioning on the condition numbers of a linear systems.

3.3 Small Rank Modification and Additive Preconditioning

In pursuing alternative preconditioning method, we are turning attention to the additive preconditioning techniques. Rank-one modification has been used for a long time in matrix computations. Similar additive process with small rank matrix also have some nice features to be explored. We will first explore the impact on conditioning

problem with small rank modification. Then we will show the recent advances on additive preconditioning methods. We present algorithms using APPs of rank one for inverse iteration and finish this section with some numerical results.

3.3.1 Small Rank Modification

The rank one modifications to a matrix has been widely studied in matrix computation for many years. We have tested that the condition of a matrix, especially for an ill-conditioned matrix is usually improved with a random rank one modification.

Let $C = A + \sigma uu^T$, where A is a diagonal matrix $D = \text{diag}(d_i)$ of size $n \times n$ and $d_k \leq d_{k+1}$, u is vector of size $n \times 1$. Let the eigenvalues of C be ordered as λ_1,λ_2,...,λ_n that $|\lambda_i| \leq |\lambda_{i+1}|$. Then we have:

i. if $\sigma > 0$, $\quad d_i \leq \lambda_i \leq d_{i+1}$, $\quad$ i = 1,2,...,n-1,

$$d_n \leq \lambda_n \leq d_n + \sigma;$$

ii. if $\sigma < 0$, $\quad d_i \leq \lambda_{i+1} \leq d_{i+1}$, $\quad$ i = 1,2,...,n-1,

$$d_1 + \sigma \leq \lambda_n \leq d_1.$$

This will give us the bounds on each of the eigenvalues of C [W65].

For an $n \times n$ matrix A with eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_n$, u and v are n-dimensional column vectors such that either u is an eigenvector of A or v is a left eigenvector of A, associated with eigenvalue λ_1. If $\alpha \epsilon [0, 1]$, then the eigenvalues of the matrix

$\alpha A + (1 - \alpha)uv^T$

are $\alpha\lambda_1 + (1 - \alpha)v^T u, \alpha\lambda_2, ..., \alpha\lambda_n$ [DZ07].

3.3.2 Inverse of the Rank Modification of a Matrix

When we use the inverse Rayleigh Quotient iteration, it is most critical at step k to compute the vector $(A - \mu I)^{-1}\omega = y_{k-1}$ where the $(A - \mu I)$ is close to singular matrix.

At this stage we can shift to a preconditioned matrix $C = A - UV^T$ and then apply the Sherman-Morrison-Woodbury inversion formula in order to compute

$$A^{-1} = (C - UV^T)^{-1} = C^{-1} + C^{-1}UG^{-1}V^HC^{-1},\ G = I - V^HC^{-1}U.$$

For random and properly scaled $n \times r$ matrices U and V and ill conditioned matrix A having a positive numerical nullity r we can expect to decrease the condition number in additive preprocessing $C \to C$, that is, in the transition to the matrix C.

3.4 Some results with Additive Preconditioners of rank one

Specifying our algorithms, we write $||\cdot||_q$ for $q = 2$ or $q = F$ to denote the 2-norm or the Frobenius norm of a matrix, respectively. We call a matrix M normalized if $||M||_2 = 1$. Actually in our algorithms we only need weak normalization, such that $||M||_2$ is neither large nor small. We employ error-free scaling by $\sigma(\tilde{\lambda})$, the powers of two. Equations (3.2) and (3.3) below define our Approaches 1 and 2, respectively.

$$C_1(\lambda) = \frac{1}{\sigma(\lambda)}A(\lambda) + UV^H,\ \ C_2(\lambda) = \frac{1}{\sigma(\lambda)}A(\lambda) + YV^H, \tag{3.1}$$

$$f_1(A(\lambda), Y) = (C_1(\lambda)^{-1} + C_1(\lambda)^{-1}UG^{-1}V^HC_1(\lambda)^{-1})Y, \tag{3.2}$$

$$f_2(A(\lambda), Y) = C_2(\lambda)^{-1}Y. \tag{3.3}$$

Algorithm 3.4. Inverse iteration with APPs of rank one

INPUT: *a matrix M, a crude approximation $\tilde{\lambda}$ to its simple eigenvalue λ, a small positive tolerance value τ, a small positive integer ν (e.g., $\nu = 1$), the assignment $q = 2$ or $q = F$, and a Subroutine LIN·SOLVE for solving a non-singular and well conditioned linear system of equations (e.g., based on PLU factorization or the Conjugate Gradient method).*

OUTPUT: *either FAILURE with a probability near 0 or an approximate eigenpair*

$(\lambda_{final}, \mathbf{y}_{final})$ *of the matrix* M *such that* $||A(\lambda_{final})\mathbf{y}_{final}||_2 \le \tau||A(\lambda)||_q$ *where* $A(\lambda) = \lambda I - M$.

INITIALIZATION: *Fix an integer* $g = 1$ *or* $g = 2$. *Write* $A(\lambda) = \lambda I - M$. *Set* $COUNTER \longleftarrow 0$, $\phi \leftarrow 0$, *and* $C_g(\tilde{\lambda}) \longleftarrow A(\tilde{\lambda})$. *Fix or randomly generate a triple of normalized vectors* $\mathbf{u}$, $\mathbf{v}$, *and* $\mathbf{y}$.

COMPUTATIONS:

1. *If* $COUNTER > \nu$, *output FAILURE and stop. Otherwise apply Subroutine LIN·SOLVE to compute the vector* $\mathbf{z} = (C_g(\tilde{\lambda}))^{-1}\mathbf{y}$ *for* $\phi = 0$ *or* $\mathbf{z} = f_g(A(\lambda), \mathbf{y})$ *for* $\phi = 1$ *where* $f_g(A(\lambda), \mathbf{y})$ *is defined by equations (3.1)–(3.3) for* $U = \mathbf{u}$, $V = \mathbf{v}$, *and* $Y = \mathbf{y}$.
2. *If this application fails (that is, if the matrix* $C_g(\tilde{\lambda})$ *is singular or ill conditioned), then set* $\phi \leftarrow 1$, *compute a crude approximation* $\sigma(\tilde{\lambda})$ *by a power of two to the norm* $||A(\tilde{\lambda})||_q$, *generate a triple of normalized random vectors* $\mathbf{u}$, $\mathbf{v}$, *and* $\mathbf{y}$, *set* $COUNTER \longleftarrow COUNTER + 1$, *and go to Stage 1.*
3. *Set* $COUNTER \longleftarrow 0$. *Compute the vector* $\mathbf{x} = \mathbf{z}/||\mathbf{z}||_2$.
4. *Compute the Rayleigh quotient* $\gamma = \mathbf{x}^H M\mathbf{x}$.
5. *If* $||A(\gamma)\mathbf{x}||_2 \le \tau||A(\gamma)||_F$ *(that is, if the residual norm is small enough), output* $\lambda_{final} = \gamma$, $\mathbf{y}_{final} = \mathbf{x}$ *and stop. Otherwise set* $\tilde{\lambda} \longleftarrow \gamma$ *and* $\mathbf{y} \longleftarrow \mathbf{x}$ *and go to Stage 1.*

In Tables 3.1 and 3.2 we show the numbers of iterations required for the convergence of the IPI and Algorithm 3.4. We display the average (mean) values and standard deviations in 200 tests with $n \times n$ matrices $A = \lambda I - M$ for $M = G^{-1}TG$, $n = 64$ and $n = 100$, G being either a random matrix or the Q-factor in the QR factorization of a random matrix, and T from one of the four following matrix classes:

Table 3.1: Iteration count for IPI and Algorithm 3.4 with unitary matrix G

Matrix Classes	n	Algorithm 3.4		IPI	
		iter	std dev	iter	std dev
$T = D_r$	64	4.74	1.145	4.93	1.242
	100	4.71	1.277	4.88	1.299
$T = D_c$	64	5.67	1.415	5.61	1.396
	100	5.67	1.461	5.62	1.321
$T =$	64	4.94	1.230	5.01	1.341
$D_r + \mathbf{e}_1\mathbf{v}^T + \mathbf{u}\mathbf{e}_n^T$	100	4.75	1.176	4.75	1.260
$T = D_r + \mathbf{u}\mathbf{v}^T$	64	5.77	1.668	5.95	1.808
	100	5.54	1.445	5.67	1.553

1. $T = D_r$ is a real diagonal matrix with random entries in the closed line interval $[0, 10]$.

2. $T = D_c$ is a complex diagonal matrix whose entries have random absolute values in the line interval $[0, 10]$ and random arguments in the semi-open line interval $[0, 2\pi)$.

3. $T = D_r + \mathbf{e}_1\mathbf{v}^T + \mathbf{u}\mathbf{e}_n^T$ is an arrow-head matrix, D_r is a matrix of class 1, and the vectors $\mathbf{u}$ and $\mathbf{v}$ have random entries in the closed line interval $[0, 10]$.

4. $T = D_r + \mathbf{u}\mathbf{v}^T$, D_r and $\mathbf{v}$ are as in matrix class 3, and the vector $\mathbf{u}$ has random coordinates in the closed line interval $[0, 1]$.

Table 3.2: Iteration count for IPI and Algorithm 3.4 with random matrices G

Matrix Classes	n	Algorithm 3.4		IPI	
		iter	std dev	iter	std dev
$T = D_r$	64	5.36	2.532	5.36	2.520
	100	4.88	2.509	4.86	2.452
$T = D_c$	64	5.76	1.716	5.71	1.516
	100	5.59	1.401	5.64	1.497
$T =$	64	5.09	1.621	5.03	1.605
$D_r + \mathbf{e}_1\mathbf{v}^T + \mathbf{u}\mathbf{e}_n^T$	100	4.72	1.473	4.67	1.467
$T = D_r + \mathbf{u}\mathbf{v}^T$	64	5.550	1.907	5.550	1.872
	100	5.660	2.118	5.555	1.992

3.5 Null-space solving with Additive Preconditioners

Given a matrix A, we seek its null vectors and its *null matrix basis*, that is, a matrix whose columns form a basis for the null space $N(A)$. Hereafter we use the abbreviations *nmb* and $\mathrm{nmb}(A)$. We link our subject to some other fundamental matrix computations.

We will show how we use APPs for null-space solving. We write $C = A + UV^H$ and $r = rank(UV^H)$, "$\Longrightarrow$" for "implies", and "$\Longleftrightarrow$" for "if and only if.".

Assuming that A and C are $m \times n$ matrices and $\operatorname{rank} C = n \le m$, we have

$$N(A) \subseteq \mathrm{range}(C^+U),$$

$$\{r = \operatorname{nul} A\} \Longleftrightarrow \{N(A) = \mathrm{range}(C^+U)\} \Longleftrightarrow \{AC^+U = 0\},$$

$$\{X \text{ is an } \mathrm{nmb}(AC^+U)\} \Longrightarrow \{range(C^+UX) \text{ is an } \mathrm{nmb}(A)\}.$$

Theorem 3.1. *Suppose $m \ge n$ and for an $m \times n$ matrix A of a rank ρ and a pair of two matrices U of size $m \times r$ and V of size $n \times r$, the matrix $C = A + UV^H$ has full rank n. Then*

$$r \ge \operatorname{rank} U \ge n - \rho = \operatorname{rnul} A = \operatorname{nul} A, \tag{3.4}$$

$$N(A) \subseteq \text{range}(C^+U). \tag{3.5}$$

Furthermore if

$$r = \text{rank}\, U = n - \rho = \text{rnul}\, A = \text{nul}\, A, \tag{3.6}$$

then

$$C^+U \text{ is an nmb}(A), \tag{3.7}$$

$$V^H C^+ U = I_r. \tag{3.8}$$

Proof. Bound (3.4) follows because $\text{rank}(B + C) \leq \text{rank}\, B + \text{rank}\, C$. If $\mathbf{y} \in N(A)$, then $C\mathbf{y} = (A + UV^H)\mathbf{y} = UV^H\mathbf{y}$, and therefore

$$\mathbf{y} = C^+U(V^H\mathbf{y}). \tag{3.9}$$

This proves (3.5).

(3.7) immediately follows from (3.5) and (3.6).

To prove (3.8), pre-multiply equation (3.9) by V^H, recall equation (3.7), and deduce that $(V^HC^+U - I_r)V^HC^+U = 0$. Now (3.8) follows unless the matrix V^HC^+U is singular, but if it is, then $V^HC^+U\mathbf{z} = \mathbf{0}$ for some nonzero vector $\mathbf{z}$. Let us write $\mathbf{w} = C^+U\mathbf{z}$, so that $V^H\mathbf{w} = \mathbf{0}$ and $\mathbf{w} \in \text{range}(C^+U) = N(A)$. It follows that $A\mathbf{w} = \mathbf{0}$, and therefore, $C\mathbf{w} = A\mathbf{w} + UV^H\mathbf{w} = \mathbf{0}$. Now recall that the matrix C has full rank and conclude that $\mathbf{w} = \mathbf{0}$. Consequently, $\mathbf{z} = \mathbf{0}$ because the matrix C^+U has full rank. □

Corollary 3.1. *Under the assumptions of Theorem 3.1 let equations (3.4) and (3.5) hold. Then C^+UX is an nmb(A) if X is an nmb(AC^+U).*

To summarize, C^+U is an nmb(A) if, under the assumptions of Theorem 3.1, we have $r = \text{nul}\, A$. If, however, $r > \text{nul}\, A$, then $N(A) \subset \text{range}(C^+U)$, and $N(A) = \text{range}(C^+UX)$ if X is an nmb(AC^+U).

We presented four algorithms in [PY07] on using additive preconditioner for null space solving which will not be discussed further from here.

4 Multiplicative Preconditioning that Stabilizes Gaussian and Block Gaussian Elimination(GENP)

Multiplicative Preconditioning method is explored to stabilize Gaussian and Block Gaussian Elimination. We will provided some basic definitions, followed by discussion of the multiplicative Preconditioning. Then Gaussian elimination is discussed and the results of numerical tests are presented at the end.

4.1 Some Definitions and GENP

4.1.1 Some Definitions

Except for using unitary circulant matrices in Sections 4.4 and 4.6.2, we assume computations in the field $\mathbb{R}$ of only real numbers, but the extension to the case of the complex field $\mathbb{C}$ is quite straightforward. Hereafter "flop" stands for "arithmetic operation", " i.i.d." stands for "independent identically distributed", and "Gaussian matrix" stands for "standard Gaussian random matrix" (cf. Definition 4.1). The concepts "large", "small", "near", "closely approximate", "ill-conditioned" and "well-conditioned" are quantified in the context. By saying "expect" and "likely" we mean "with probability 1 or close to 1".

Next we recall and extend some customary definitions of matrix computations [GL96], [S98].

In additional to notations in section 2.1 we will use the following notations: $\mathbb{R}^{m\times n}$ is the class of real $m \times n$ matrices $A = (a_{i,j})_{i,j}^{m,n}$.

A real matrix Q is *orthogonal* if $Q^TQ = I$ or $QQ^T = I$. $(Q, R) = (Q(A), R(A))$ for an $m \times n$ matrix A of rank n denotes a unique pair of orthogonal $m \times n$ and upper triangular $n \times n$ matrices such that $A = QR$ and all diagonal entries of the matrix R

are positive [GL13, Theorem 5.2.3].

A^+ denotes the Moore–Penrose pseudo-inverse of an $m \times n$ matrix A, and

$$A = S_A \Sigma_A T_A^T \tag{4.1}$$

denotes its SVD where $S_A^T S_A = S_A S_A^T = I_m$, $T_A^T T_A = T_A T_A^T = I_n$, $\Sigma_A = \text{diag}(\sigma_j(A))_j$, and $\sigma_j = \sigma_j(A)$ is the jth largest singular value of A. If a matrix A has full column rank ρ, then

$$||A^+|| = 1/\sigma_\rho(A). \tag{4.2}$$

A^{+T} stands for $(A^+)^T = (A^T)^+$, A_s^T for $(A_s)^T$, and A_s^+ for $(A_s)^+$ where s can denote a scalar, a matrix, or a pair of such objects, e.g., $A_{k,l}^T$ stands for $(A_{k,l})^T$.

$\kappa(A) = \frac{\sigma_1(A)}{\sigma_\rho(A)} = ||A||\ ||A^+||$ is the condition number of an $m \times n$ matrix A of a rank ρ. Such matrix is *ill-conditioned* if the ratio $\sigma_1(A)/\sigma_\rho(A)$ is large. If the ratio is reasonably bounded, then the matrix is *well-conditioned*. An $m \times n$ matrix A has a *numerical rank* $r = \text{nrank}(A) \le \rho = \text{rank}(A)$ if the ratios $\sigma_j(A)/||A||$ are small for $j > r$ but not for $j \le r$.

The following concepts cover all rectangular matrices, but we need them just in the case of square matrices, whose sets of leading blocks include the matrices themselves. A matrix is *strongly nonsingular* if all its leading blocks are nonsingular. Such a matrix is *strongly well-conditioned* if all its leading blocks are well-conditioned.

4.1.2 Block Gaussian elimination and GENP

For a nonsingular 2×2 block matrix $A = \begin{pmatrix} B & C \\ D & E \end{pmatrix}$ of size $n \times n$ with nonsingular $k \times k$ *pivot block* $B = A^{(k)}$, define $S = S(A^{(k)}, A) = E - DB^{-1}C$, the *Schur complement* of $A^{(k)}$ in A, and the block factorizations,

$$A = \begin{pmatrix} I_k & O_{k,r} \\ DB^{-1} & I_r \end{pmatrix} \begin{pmatrix} B & O_{k,r} \\ O_{r,k} & S \end{pmatrix} \begin{pmatrix} I_k & B^{-1}C \\ O_{k,r} & I_r \end{pmatrix} \tag{4.3}$$

and

$$A^{-1} = \begin{pmatrix} I_k & -B^{-1}C \\ O_{k,r} & I_r \end{pmatrix} \begin{pmatrix} B^{-1} & O_{k,r} \\ O_{r,k} & S^{-1} \end{pmatrix} \begin{pmatrix} I_k & O_{k,r} \\ -DB^{-1} & I_r \end{pmatrix}. \tag{4.4}$$

We verify readily that S^{-1} is the $(n-k) \times (n-k)$ trailing (that is, southeastern) block of the inverse matrix A^{-1}, and so the Schur complement S is nonsingular since the matrix A is nonsingular.

Factorization (4.4) reduces the inversion of the matrix A to the inversion of the leading block B and its Schur complement S, and we can recursively reduce the task to the case of the leading blocks and Schur complements of decreasing sizes as long as the leading blocks are nonsingular. After sufficiently many recursive steps of this process of block Gaussian elimination, we only need to invert matrices of small sizes, and then we can stop the process and apply a selected black box inversion algorithm.

In $\lceil \log_2(n) \rceil$ recursive steps all pivot blocks and all other matrices involved into the resulting factorization turn into scalars, all matrix multiplications and inversions turn into scalar multiplications and divisions, and we arrive at a *complete recursive factorization* of the matrix A. If $k = 1$ at all recursive steps, then the complete recursive factorization (4.4) defines GENP and can be applied to computing the inverse A^{-1} or the solution $\mathbf{y} = A^{-1}\mathbf{b}$ to a linear system $A\mathbf{y} = \mathbf{b}$.

Actually, however, any complete recursive factorizations turns into GENP up to the order in which we consider its steps. This follows because at most $n-1$ distinct Schur complements $S = S(A^{(k)}, A)$ for $k = 1, \dots, n-1$ are involved in all recursive block factorization processes for $n \times n$ matrices A, and so we arrive at the same Schur complement in a fixed position via GENP and via any other recursive block factor-

ization (4.3). Hence we can interpret factorization step (4.3) as the block elimination of the first k columns of the matrix A, which produces the matrix $S = S(A^{(k)}, A)$. If the dimensions $d_1, \dots, d_r$ and $\bar{d}_1, \dots, \bar{d}_{\bar{r}}$ of the pivot blocks in two block elimination processes sum to the same integer k, that is, if $k = d_1 + \cdots + d_r = \bar{d}_1 + \cdots + \bar{d}_{\bar{r}}$, then both processes produce the same Schur complement $S = S(A^{(k)}, A)$. The following results extend this observation.

Theorem 4.1. *In the recursive block factorization process based on (4.3), every diagonal block of every block diagonal factor is either a leading block of the input matrix A or the Schur complement $S(A^{(h)}, A^{(k)})$ for some integers h and k such that $0 < h < k \le n$ and $S(A^{(h)}, A^{(k)}) = (S(A^{(h)}, A))^{(h)}$.*

Corollary 4.1. *The recursive block factorization process based on equation (4.3) can be completed by involving no singular pivot blocks (and in particular no pivot elements vanish) if and only if the input matrix A is strongly nonsingular.*

Proof. Combine Theorem 4.1 with the equation $\det A = (\det B) \det S$, implied by (4.3). □

The following theorem bounds the norms of all pivot blocks and their inverses and hence bounds the condition numbers of the blocks, that is, precisely the quantities responsible for safe numerical performance of block Gaussian elimination and GENP.

Theorem 4.2. *(Cf. [PQZ13, Theorem 5.1].) Assume GENP or block Gaussian elimination applied to an $n \times n$ matrix A and write $N = ||A||$ and $N_- = \max_{j=1}^{n} ||(A^{(j)})^{-1}||$, and so $N_- N \ge ||A||\ ||A^{-1}|| \ge 1$. Then the absolute values of all pivot elements of GENP and the norms of all pivot blocks of block Gaussian elimination do not exceed $N_+ = N + N_- N^2$, while the absolute values of the reciprocals of these elements and the norms of the inverses of the blocks do not exceed N_-.*

Proof. Observe that the inverse S^{-1} of the Schur complement S in (4.3) is the south-eastern block of the inverse A^{-1} and obtain $||B|| \le N$, $||B^{-1}|| \le N_-$, and $||S^{-1}|| \le ||A^{-1}|| \le N_-$. Moreover $||S|| \le N + N_- N^2$, due to (4.3). Now the claimed bound follows from Theorem 4.1. □

Remark 4.1. *By virtue of Theorem 4.2 the norms of the inverses of all pivot blocks involved into a complete (and hence also into any incomplete) recursive factorization of a strongly nonsingular matrix A are at most N_-. We have a reasonable upper bound on N_- if the matrix A is strongly well-conditioned as well. Then in view of Theorem 4.13 the inversion of all pivot blocks is numerically safe, and we say that* GENP is locally safe *for the matrix A.*

Remark 4.2. *In the recursive factorizations above only the factors of the leading blocks and the Schur complements can contribute to the magnification of any input perturbation. Namely at most $\lceil \log_2(n) \rceil$ such factors can contribute to the norm of each of the output triangular or block triangular factors L and U. This implies the moderately large worst case upper bound $(N_+ N_-)^{\log_2(n)}$ on their norms, which is overly pessimistic according to our tests.*

Remark 4.3. *Our study in this and the next two sections can be extended readily to the cases of GENP and block Gaussian elimination applied to rectangular and possibly rank deficient matrices and to under- and over-determined and possibly rank deficient linear systems of equations. Recursive factorization and elimination can be completed and are numerically safe when they are applied to any strongly nonsingular and strongly well-conditioned leading block of the input matrix, in particular to the input matrix itself if it is strongly nonsingular and strongly well-conditioned.*

4.2 Traditional Multiplicative Preconditioning

The nomenclature *preconditioning* refers to transforming a linear system of equations into a system with more favorable properties for iterative solution, e.g., the new linear system can have smaller condition number. The solution to the original system is either preserved or can be easily computed based on that of the new system. Multiplicative preconditioning has been widely adopted for this purpose. In particular a linear system $Ax = b$ can be modified into $M^{-1}Ax = M^{-1}b$. Here the matrix M^{-1} is called a preconditioner or more specifically a *multiplicative preconditioner*. More generally the preconditioning take the form of $M^{-1}AN^{-1}y = M^{-1}b$, $N^{-1}y = x$. The purpose of preconditioning is to have a better conditioned matrix $M^{-1}A$ (the closer to identity the better), with which linear system is easier to solve and which is less prone to numerical stability problems. Similarly if the matrix M is Hermitian and positive definite, then the modified linear system takes the form $L^{-1}AL^{-H}y = L^{-1}b, x = L^{-1}y$, where $M = LL^H$. The matrix L could be the Hermitian square root of M or Cholesky factor of it. When $M = diag(A)$, then this becomes the Jacobi classical iteration. Let $D = D(A)$ denote the diagonal matrix made up from the diagonal of A, and transform the linear system $M^{-1}Ax = M^{-1}b$ into $x = D^{-1}(D - A)x + D^{-1}b$.

Algorithm 4.1. Jacobi preconditioned Iteration

INPUT: *A matrix A, a vector b, an initial guess of* $\mathbf{x}_0$ *for* $A * y = b$*, a positive value* τ*, a positive integer* γ*.*

OUTPUT: *either FAILURE or* $\mathbf{x}_{final}$ *that satisfy* $||A(\mathbf{x})_{final} - b||_2 \leq t$*.*

INITIALIZATION: $COUNT \longleftarrow 0$*.*

COMPUTATIONS:

1. $\mathbf{x}_{count} \longleftarrow D^{-1}(D - A)x_{count-1} + D^{-1}b$

2. *produce FAILURE if* $COUNT > \gamma$ *or repeat until* $||A(\mathbf{x})_{count} - b||_2 \leq \tau$ *then output* $\mathbf{x}_{final} = \mathbf{x}_{count}$.

Similarly if M is the lower triangular matrix made up of the lower triangular part of the matrix A, then the preconditioned method results in Gauss-Seidel iteration. These two methods are generally considered stationary iteration methods. Non-stationary methods differ from stationary methods in that the computation involves information that changes in each iteration. Conjugate Gradient method - CG, MINRES, GMRES, Lanczos, and Arnoldi iteration are the most popular ones of these type. Preconditioners are widely used in the known implementation of those iterations that will not be reviewed further here.

"It is widely recognized that preconditioning is the most critical ingredient in the development of efficient solvers for challenging problems in scientific computation, and that the importance of preconditioning is destined to increase even further" [B02].

Generally preconditioners can be divided into three categories [A97]:

1. Preconditioners designed for quite general although still special classes of matrices; e.g., matrices with nonzero diagonal entries or positive definite matrices where the iterations such as Jacobi, Gauss-Seidel, incomplete Cholesky are used.

2. Preconditioners designed for a broad but special class of underlying problems; e.g., elliptic PDEs, where multigrid and domain decomposition preconditioners are used.

3. Preconditioners designed for a specific matrix or underlying problem; e.g., the transport equation, where DSA preconditioner is used.

It was found that a preconditioner may work well for one type of matrix but may be not suitable for another. There were a lot of effort in finding a general purpose

multiplicative preconditioner, but this is still a research challenge. There are two main requirements in the design of preconditioners [B02]:

- The preconditioned system should be easy to solve
- It should be inexpensive to construct and apply the preconditioner.

Incomplete factorization methods have been introduced for a quasi-LU factorization of M such that $M = \bar{L}\bar{U}$, where $\bar{L}$ and $\bar{U}$ are the incomplete LU factors that discard part of the fill-in not near the diagonal in the factorization process. Although it is a popular approach used in iterative methods, but it has limitations such as potential instability, lack of scalability and difficulty for to parallelization.

Sparse approximate inverse would be the algorithm of choice where $M \cong A$. Generally it is expensive to invert such a matrix M, and there are stability issues that may cause failure even for a general SPD matrix [B02].

4.3 Stabilization of GENP and block Gaussian elimination with Mutilicative Preconditioner

4.3.1 Preconditioning of GENP and block Gaussian Elimination

Preprocessing $A \to FAH$ for a pair of nonsingular matrices F and H, one of which can be the identity matrix I, reduces the inversion of a matrix A to the inversion of a the product FAH, and similarly for the solution of a linear system of equations.

Fact 4.1. *Assume three nonsingular matrices F, A, and H and a vector $\mathbf{b}$. Then $A^{-1} = H(AH)^{-1}$, $A^{-1} = (FA)^{-1}F$, $A^{-1} = H(FAH)^{-1}F$. Moreover, if $A\mathbf{x} = \mathbf{b}$, then $AH\mathbf{y} = \mathbf{b}$, $FA\mathbf{x} = F\mathbf{b}$, and $FAH\mathbf{y} = F\mathbf{b}$, $\mathbf{x} = H\mathbf{y}$.*

Remark 4.1 in section 4.1 motivates the choice of the multipliers F and H for which the matrix FAH is strongly nonsingular and strongly well-conditioned. This is

likely to occur already if one of the multipliers F and H is the identity matrix and another one is a Gaussian random matrix. The studies of pre-multiplication by F and post-multiplication by H are similar, and so we only prove the latter claim in the case of post-multiplication. Later we complete our proof which involves the norms of the inverses of the matrices $(AH)_{k,k} = A_{k,n}H_{n,k}$ for $k = 1, \dots, r$, that we estimate in this section assuming nonrandom multipliers H. We begin with two simple lemmas.

Lemma 4.1. *If S and T are square orthogonal matrices, then $\sigma_j(SA) = \sigma_j(AT) = \sigma_j(A)$ for all j.*

Lemma 4.2. *Suppose $\Sigma = \mathrm{diag}(\sigma_i)_{i=1}^n$, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$, and $H \in \mathbb{R}^{n\times r}$. Then*

$$\sigma_j(\Sigma H) \geq \sigma_j(H)\sigma_n \text{for all } j.$$

If also $\sigma_n > 0$, then

$$\mathrm{rank}(\Sigma H) = \mathrm{rank}(H).$$

We also need the following basic results (cf. [GL13, Corollary 8.6.3]).

Theorem 4.3. *If A_0 is a submatrix of a matrix A, then $\sigma_j(A) \geq \sigma_j(A_0)$ for all j.*

Theorem 4.4. *Suppose $r + l \leq n \leq m$, $l \geq 0$, $A \in \mathbb{R}^{m\times n}$, $\mathrm{rank}(A_{m,r}) = r$ and $\mathrm{rank}(A_{m,r+l}) = r + l$. Then $||A^+_{m,r}|| \leq ||A^+_{m,r+l}||$.*

The following theorem will enable us to estimate the norm $||(AH)^+||$.

Theorem 4.5. *Suppose $A \in \mathbb{R}^{n\times n}$, $H \in \mathbb{R}^{n\times r}$, $\mathrm{rank}(A) = n \geq r$, $A = S_A\Sigma_A T_A^T$ is SVD (cf. (4.1)), and $\widehat{H} = T_A^T H$. Then*

$$\sigma_j(AH) \geq \sigma_l(A)\ \sigma_j(\widehat{H}_{l,r}) \text{ for all } l \leq n \text{ and all } j. \tag{4.5}$$

Proof. Note that $AH = S_A \Sigma_A T_A^T H$, and so $\sigma_j(AH) = \sigma_j(\Sigma_A T_A^T H) = \sigma_j(\Sigma_A \widehat{H})$ for all j by virtue of Lemma 4.1, because S_A is a square orthogonal matrix. Moreover it follows from Theorem 4.3 that $\sigma_j(\Sigma_A \widehat{H}) \geq \sigma_j(\Sigma_{l,A} \widehat{H}_{l,r})$ for all $l \leq n$. Combine this bound with the latter equations and apply Lemma 4.2. □

Corollary 4.2. *Keep the assumptions of Theorem 4.5. Then*

(i) $\sigma_r(AH) \geq \sigma_\rho(A)\sigma_r(\widehat{H}_{n,r}) = \sigma_r(\widehat{H}_{n,r})/||A^+||$,

(ii) $||(AH)^+|| \leq ||A^+||\ ||\widehat{H}^+_{n,r}||$ *if* $\operatorname{rank}(AH) = \operatorname{rank}(\widehat{H}_{n,r}) = r$.

Proof. Substitute $j = r$ and $l = n$ into bound (4.5), recall (4.2), and obtain part (i). If $\operatorname{rank}(AH) = \operatorname{rank}(\widehat{H}_{l,r}) = r$, then apply (4.2) to obtain that $\sigma_r(AH) = 1/||(AH)^+||$ and $\sigma_r(\widehat{H}_{l,r}) = 1/||\widehat{H}^+_{l,r}||$. Substitute these equations into part (i) and obtain part (ii). □

Let us extend the estimates of Theorem 4.5 to the leading blocks of a matrix product.

Corollary 4.3. *Keep the assumptions of Theorem 4.5 and also suppose that the matrices* $(AH)_{k,k}$ *and* $\widehat{H}_{n,k}$ *have full rank* k *for a positive integer* $k \leq n$. *Then*

$$||(AH)^+_{k,k}|| \leq ||\widehat{H}^+_{n,k}||\ ||A^+_{k,n}|| \leq ||\widehat{H}^+_{n,k}||\ ||A^+||.$$

Proof. Note that $(AH)_{k,k} = A_{k,n} H_{n,k}$ and that the matrix $A_{k,n}$ has full rank. Apply Corollary 4.2 for A and H replaced by $A_{k,n}$ and $H_{n,k}$, respectively, and obtain that $||(AH)^+_{k,k}|| \leq ||\widehat{H}^+_{k,n}||\ ||A^+_{n,k}||$. Combine (4.2) and Theorem 4.4 and deduce that $||A^+_{n,k}|| \leq ||A^+||$. Combine the two latter inequalities to complete the proof of part (i). Similarly prove part (ii). □

Fact 4.1, Corollary 4.1 and Theorem 4.2 together imply the following result.

Corollary 4.4. *Suppose that $A \in \mathbb{R}^{n\times n}$, $H \in \mathbb{R}^{n\times r}$, $r \le n = \mathrm{rank}(A)$, and the matrices $(AH)_{k,k}$ are strongly nonsingular and strongly well-conditioned for $k = 1, \dots, r$. Then GENP and block Gaussian elimination are locally safe for the matrix product AH (see Remark 4.1 on the concept "locally safe").*

Definition 4.1. *A matrix is said to be* standard Gaussian random *(hereafter we say just* Gaussian*) if it is filled with i.i.d. Gaussian random variables having mean* 0 *and variance* 1*.*

Theorem 4.6. *A Gaussian matrix G is strongly nonsingular with probability 1.*

Proof. Assume that the $j \times j$ leading submatrix $G^{(j)}$ of a $k \times l$ Gaussian matrix G is singular for some positive integer $j \le h = \min\{k, l\}$, that is, $\det(G^{(j)}) = 0$. Since $\det(G^{(j)})$ is a polynomial in the entries of the Gaussian matrix $G^{(j)}$, such matrices form an algebraic variety of a lower dimension in the linear space $\mathbb{R}^{j^2}$. (V is an algebraic variety of a dimension $d \le N$ in the space $\mathbb{R}^N$ if it is defined by $N - d$ polynomial equations and cannot be defined by fewer equations.) Clearly, Lebesgue (uniform) and Gaussian measures of such a variety equal 0, being absolutely continuous with respect to one another. Hence these measures of the union of h such matrices are also 0. □

Theorem 4.7. *Assume a nonsingular $n \times n$ matrix A and an $n \times k$ Gaussian matrix $H_{n,k}$. Then the product $A_{k,n}H_{n,k}$ is nonsingular with probability 1.*

Proof. $\det(A_{k,n}H_{n,k})$ is a polynomials in the entries of the Gaussian matrix $H_{n,k}$. Such a polynomial vanishes with probability 0 unless it vanishes identically in $H_{n,k}$, but the matrix $A_{k,n}A_{k,n}^T$ is positive definite, and so $\det(A_{k,n}H_{n,k}) > 0$ for $H_{n,k} = A_{k,n}^T$. □

Definition 4.2. *$\nu_{j,m,n}$ denotes the random variables $\sigma_j(G)$ for a Gaussian $m\times n$ matrix G and all j, while $\nu_{m,n}$, $\nu_{F,m,n}$, $\nu^+_{m,n}$, and $\kappa_{m,n}$ denote the random variables $||G||$, $||G||_F$, $||G^+||$, and $\kappa(G) = ||G||\ ||G^+||$, respectively.*

Note that $\nu_{j,n,m} = \nu_{j,m,n}$, $\nu_{n,m} = \nu_{m,n}$, $\nu^+_{n,m} = \nu^+_{m,n}$, and $\kappa_{n,m} = \kappa_{m,n}$.

Theorem 4.8. *(Cf. [DS01, Theorem II.7].) Suppose* $h = \max\{m, n\}$, $t \geq 0$, *and* $z \geq 2\sqrt{h}$. *Then* Probability $\{\nu_{m,n} > z\} \leq \exp(-(z - 2\sqrt{h})^2/2\}$ *and* Probability $\{\nu_{m,n} > t + \sqrt{m} + \sqrt{n}\} \leq \exp(-t^2/2)$.

Theorem 4.9. *(Cf. [CD05, Proof of Lemma 4.1].) Suppose* $m \geq n \geq 2$, *and* $x > 0$ *and write* $\Gamma(x) = \int_0^\infty \exp(-t)t^{x-1}dt$ *and* $\zeta(t) = t^{m-1}m^{m/2}2^{(2-m)/2}\exp(-mt^2/2)/\Gamma(m/2)$. *Then* Probability $\{\nu^+_{m,n} \geq m/x^2\} < \frac{x^{m-n+1}}{\Gamma(m-n+2)}$.

The following condition estimates from [CD05, Theorem 4.5] are quite tight for large values x, but for $n \geq 2$ even tighter estimates (although more involved) can be found in [ES05]. (See [D88] and [E88] on the early study.)

Theorem 4.10. *If* $m \geq n \geq 2$, *then*

$$\text{Probability } \{\kappa_{m,n}m/(m-n+1) > x\} \leq \frac{1}{2\pi}(6.414/x)^{m-n+1}$$

for $x \geq m - n + 1$, *while* $\kappa_{m,1} = 1$ *with probability 1.*

Corollary 4.5. *A Gaussian matrix is expected to be strongly well-conditioned.*

The main result of this section is the Corollary 4.7, which supports application of GENP and block Gaussian elimination to the product AH of a nonsingular matrix A and a Gaussian matrix H.

We need the following simple basic lemma.

Lemma 4.3. *Suppose* H *is a Gaussian matrix,* S *and* T *are orthogonal matrices,* $H \in \mathbb{R}^{m\times n}$, $S \in \mathbb{R}^{k\times m}$, *and* $T \in \mathbb{R}^{n\times k}$ *for some* k, m, *and* n. *Then* SH *and* HT *are Gaussian matrices.*

Corollary 4.6. *Suppose A is a nonsingular $n \times n$ matrix, H is an $n \times k$ Gaussian matrix, for $0 < k \leq n$, and $\nu_{g,h}$ and $\nu^+_{g,h}$ are the random values of Definition 4.2. Then*

(i) the matrix $(AH)_{k,k})$ is nonsingular with probability 1,

(ii) $||(AH)_{k,k}|| \leq \nu_{n,k}||A_{k,n}|| \leq \nu_{n,k}||A||$, *and*

(iii) $||(AH)^+_{k,k}|| \leq \nu^+_{n,k}||A^+||$.

Proof. Part (i) restates Theorem 4.7. Part (ii) follows because $(AH)_{k,k} = A_{k,n}H_{n,k}$, $H_{n,k}$ is a Gaussian matrix, and $||A_{k,n}|| \leq ||A||$. Part (iii) follows from Corollary 4.3 because $\widehat{H}_{n,k}$ is a Gaussian matrix by virtue of Lemma 4.3. □

Corollary 4.7. *Suppose that the $n \times n$ matrix A is nonsingular and well-conditioned. Then the choice of Gaussian multiplier H is expected to satisfy the assumptions of Corollary 4.4.*

Proof. Recall that $(AH)_{k,k} = A_{k,n}H_{n,k}$ and hence $||(AH)_{k,k}|| = ||A_{k,n}||\nu_{n,k}$. Then combine Theorems 4.7, 4.8, and 4.9 and Corollary 4.6. □

4.4 Random structured multipliers for GENP and block Gaussian elimination

This subsection involves complex matrices. A complex matrix M is unitary if $M^HM = I$ or $MM^H = I$ where M^H denotes its Hermitian transpose, so that $M^H = M^T$ for a real matrix M.

Hereafter $\omega = \omega_n = \exp(\frac{2\pi}{n}\sqrt{-1})$ denotes an nth primitive root of unity, $\Omega = (\omega^{ij})^{n-1}_{i,j=0}$ is the matrix of the discrete Fourier transform at n points (we use the acronym *DFT*), and $\Omega^{-1} = \frac{1}{n}\Omega^H$.

An $n \times n$ circulant matrix $C = (c_{i-j \mod n})^{n-1}_{i,j=0}$ is defined by its first column $\mathbf{c} = (c_i)^{n-1}_{i=0}$.

Example 4.1. Generation of random real circulant matrices. *Generate the vector* $\mathbf{c}$ *of* n *i.i.d. random real variables in the range* $[-1, 1]$ *under the uniform probability distribution on this range. Define an* $n \times n$ *circulant matrix* C *with the first column* $\mathbf{c}$.

The following theorem links the matrices Ω and Ω^{-1} to the class of circulant matrices.

Theorem 4.11. *(Cf. [CPW74].) Let* C *denote a circulant* $n \times n$ *matrix defined by its first column* $\mathbf{c}$ *and write* $\mathbf{u} = (u_i)_{i=1}^n = \Omega\mathbf{c}$. *Then*

$$C = \Omega^{-1} \operatorname{diag}(u_j)_{j=1}^n \Omega.$$

Furthermore

$$C^{-1} = \Omega^{-1} \operatorname{diag}(1/u_j)_{j=1}^n \Omega$$

if the matrix C *is nonsingular.*

By using FFT, one can multiply the matrices Ω and $\Omega^H = \Omega^{-1}$ by a vector by using $O(n \log(n))$ flops for any n (cf., e.g., [P01, page 29]), and Theorem 4.11 extends this complexity bound to multiplication of an $n \times n$ circulant matrix and its inverses by a vector.

We need $2n^3 - n^2$ flops in order to compute the product AH of the pair of $n \times n$ matrices A and H. If, however, H is a circulant matrix, then we can compute AH by using order of $n^2 \log(n)$ flops. For a Toeplitz-like matrix A defined by its displacement generator of bounded length l, we use $O(ln \log(n))$ flops in order to compute a displacement generator of length l for the matrix AH. (See [P01] for the definition of displacement generators.) In the case of Toeplitz matrices we have $l \leq 2$ and use $O(n \log(n))$ flops. This motivates using Gaussian circulant multipliers H, that is, circulant matrices H whose first column vector is Gaussian. It has been proved in

[PSZa] that such matrices are expected to be well-conditioned, which is required for any multiplicative preconditioner.

We can define a unitary circulant matrix by its first column vector

$$\mathbf{c} = \Omega(\exp(r_i\sqrt{-1}))_{i=0}^{n-1}$$

for any set of real values $r_0, \dots, r_{n-1}$.

Example 4.2. Generation of random unitary circulant matrices.

(i) Generate a vector $\mathbf{u} = (u_j)_{j=1}^{n}$ *where* $u_j = \exp(2\pi\phi_j\sqrt{-1})$ *(and so* $|u_j| = 1$ *for all i) and where* $\phi_1, \dots, \phi_n$ *are n independent random real variables, e.g., Gaussian variables or the variables uniformly distributed in the range* $[0, 1)$.

(ii) Compute the vector $\mathbf{c} = \Omega^{-1}\mathbf{u}$*, where* Ω *denotes the* $n \times n$ *DFT matrix. Output the unitary circulant matrix C defined by its first column* $\mathbf{c}$.

Our proof that Gaussian multipliers enforce strong nonsingularity of a nonsingular matrix with probability 1 (see Theorem 4.7) has been non-trivially extended in [PZa] to the case of Gaussian circulant multipliers. Furthermore strong nonsingularity holds with probability close to 1 if we fill the first column of a multiplier F or H with i.i.d. random variables defined under the uniform probability distribution over a sufficiently large finite set (see Appendix [PSZa]).

In our tests with random input matrices, Gaussian circulant and general Gaussian multipliers have shown the same power of supporting numerically safe GENP (see Section 4.6.1), but we cannot extend our basic Lemma 4.3 and our Corollary 4.7 to the case of circulant matrices. Moreover our Theorem 4.12 and Remark 4.4 below show that, for a specific narrow class of input matrices A, GENP with these multipliers is expected to fail numerically.

Theorem 4.12. *Assume a large integer n and the $n \times n$ DFT matrix Ω, which is unitary up to scaling by $1/\sqrt{n}$.*

(i) Then application of GENP to this matrix fails numerically and

(ii) a Gaussian circulant $n \times n$ multiplier $C = \Omega^{-1} D \Omega$ with Gaussian diagonal matrix $D = \mathrm{diag}(g_j)_{j=1}^{n}$ (having i.i.d. Gaussian diagonal entries $g_1, \dots, g_n$) is not expected to fix this problem.

Proof. (i) Subtract the first row of the block $\Omega_{2,2}$ of the matrix Ω and the resulting vector from its second row. Obtain the vector $(0, \omega - 1)$ with the norm $|\omega - 1| = 2\sin(\pi/n)$. Assume that n is large and then observe that $2\sin(\pi/n) \approx 2\pi/n$ and that the value $2\pi/n$ is small, implying that $\mathrm{nrank}(\Omega_{2,2}) = 1$ because $||\Omega_{2,2}|| \geq \sqrt{2}$ for large n.

(ii) Note that $\Omega C = D\Omega$. The Gaussian variable g_1 vanishes with probability 0, and so we can assume that $g_1 \neq 0$. Multiply the first row of the block $(D\Omega)_{2,2}$ of the matrix $D\Omega$ by g_2/g_1 and subtract the resulting vector from the second row. Obtain the vector $(0, (\omega - 1)g_2)$ with the norm $|(\omega - 1)g_2| = 2|g_2 \sin(\pi/n)|$ Assume that n is large and then observe that $|(\omega - 1)g_2| \approx 2|g_2|\pi/n$ and that the variable $2|g_2|\pi/n$ is expected to be small. Hence $\mathrm{nrank}((\Omega C)_{2,2}) = \mathrm{nrank}((D\Omega)_{2,2})$ is expected to equal 1 because $||\Omega C)_{2,2}|| \leq ||\Omega_{2,2}|| \max\{g_1, g_2\}$, $||\Omega_{2,2}|| \geq \sqrt{2}$ and the random variable $\max\{|g_1|, |g_2|\}$ is not expected to be close to 0. □

Remark 4.4. *The same argument shows that Gaussian circulant multipliers C are not expected to support GENP for a bit larger class of matrices, e.g., for $A = M\Omega$ where $M = \mathrm{diag}(D_i)_{i=1}^{k}$, $D_1 = \mathrm{diag}(d_1, d_2)$, and d_1 and d_2 are two positive constants and the input size $n \times n$ is large as well as where the matrix M is strongly diagonally dominant. The reader is challenged to find out whether GENP with a Gaussian circulant preprocessor is expected to fail numerically for other classes of input matrices, in particular for any subclass of the classes of Toeplitz or Toeplitz-like matrices (cf. [P01]*

and [P15] on these classes). Another challenge is to choose a distinct random structured preprocessor for which the above problem is avoided. E.g., consider the product $\prod_{i=1}^{h} C_i$ *where* h *is a small integer exceeding 1,* C_{2j} *are circulant matrices and* C_{2j-1} *are skew-circulant (see the definition in [P01]). Toward the same goal we can apply simultaneously random structured pre- and post-multipliers* F *and* H*, defined by some i.i.d. random parameters, or the pairs of PRMB multipliers of [BBD12]. In the case of Toeplitz or Toeplitz-like input matrices* A*, the multiplications* FA *and* AH *are much less costly if the multipliers* F *and* H *are circulant matrices, skew-circulant matrices, or the products of such matrices.*

4.5 Extention to Low-rank approximation of a matrix

Suppose we seek a rank-r approximation of a matrix A that has a small numerical rank r. One can solve this problem by computing SVD of the matrix A or, at a lower cost, by computing its rank-revealing factorization [GE96], [HP92], [P00a], but using random matrix multipliers instead has some benefits [HMT11]. In this section we study the latter randomized approach. In its first subsection we recall some relevant definitions and auxiliary results.

4.5.1 Truncation of SVD. Leading and trailing singular spaces

Truncate the square orthogonal matrices S_A and T_A and the square diagonal matrix Σ_A of the SVD of (4.1), write $S_{\rho,A} = (S_A)_{m,\rho}$, $T_{\rho,A} = (T_A)_{n,\rho}$, and $\Sigma_{\rho,A} = (\Sigma_A)_{\rho,\rho} = \mathrm{diag}(\sigma_j)_{j=1}^{\rho}$, and obtain *thin SVD*

$$A = S_{\rho,A}\Sigma_{\rho,A}T_{\rho,A}^T,\ \rho = \mathrm{rank}(A). \tag{4.6}$$

Now for every integer r in the range $1 \le r \le \rho = \text{rank}(A)$, write $\Sigma_{\rho,A} = \text{diag}(\Sigma_{r,A}, \bar{\Sigma}_{A,r})$ and partition the matrices $S_{\rho,A}$ and $T_{\rho,A}$ into block columns, $S_{\rho,A} = (S_{r,A} \mid \bar{S}_{A,r})$, and $T_{\rho,A} = (T_{r,A} \mid \bar{T}_{A,r})$ where $\Sigma_{r,A} = (\Sigma_A)_{r,r} = \text{diag}(\sigma_j)_{j=1}^{r}$, $S_{r,A} = (S_A)_{m,r}$, and $T_{r,A} = (T_A)_{n,r}$. Then partition the thin SVD as follows,

$$A_r = S_{r,A}\Sigma_{r,A}T_{r,A}^T,\ \bar{A}_r = \bar{S}_{A,r}\bar{\Sigma}_{A,r}\bar{T}_{A,r}^T,\ A = A_r + \bar{A}_r \tag{4.7}$$

for $1 \le r \le \rho = \text{rank}(A)$,

and call the above decomposition the *r-truncation of thin SVD* (4.6). Note that $\bar{A}_\rho$ is an empty matrix and recall that

$$||A - A_r|| = \sigma_{r+1}(A). \tag{4.8}$$

Let $\mathbb{S}_{r,A}$ and $\mathbb{T}_{r,A}$ denote the ranges (that is, the column spans) of the matrices $S_{r,A}$ and $T_{A,r}$, respectively. If $\sigma_r > \sigma_{r+1}$, then $\mathbb{S}_{r,A}$ and $\mathbb{T}_{r,A}$ are the left and right *leading singular spaces*, respectively, associated with the r largest singular values of the matrix A. The left singular spaces of a matrix A are the right singular spaces of its transpose A^T and vice versa. All matrix bases for the singular spaces $\mathbb{S}_{r,A}$ and $\mathbb{T}_{r,A}$ are given by the matrices $S_{r,A}X$ and $T_{r,A}Y$, respectively, for nonsingular $r \times r$ matrices X and Y. The bases are orthogonal where the matrices X and Y are orthogonal.

Theorem 4.13. *[S98, Corollary 1.4.19]. Assume a pair of square matrices A (nonsingular) and E such that $||A^{-1}E|| < 1$. Then $||(A+E)^{-1}|| \le \frac{||A^{-1}||}{1-||A^{-1}E||}$ and $\frac{||(A+E)^{-1}-A^{-1}||}{||A^{-1}||} \le \frac{||A^{-1}||}{1-||A^{-1}E||}$.*

Theorem 4.14. *[S95, Theorem 5.1]. Assume a pair of $m \times n$ matrices A and $A+E$, and let the norm $||E||$ be small. Then $||Q(A+E)-Q(A)|| \le \sqrt{2}||A^+||\ ||E||_F + O(||E||_F^2)$.*

P_A denotes the *orthogonal projector* on the range of a matrix A having full column

rank,

$$P_A = A(A^TA)^{-1}A^T = AA^+ = QQ^T \text{ for } Q = Q(A). \tag{4.9}$$

Corollary 4.8. *Suppose $m \times n$ matrices A and $A + E$ have full rank. Then*

$$||P_{A+E} - P_A|| \le 2||Q(A + E) - Q(A)|| \le 2\sqrt{2}\ ||A^+||\ ||E||_F + O(||E||_F^2).$$

Proof. Clearly $P_{A+E} - P_A = Q(A + E)Q(A + E)^T - Q(A)Q(A)^T =$

$$(Q(A + E) - Q(A))Q(A + E)^T + Q(A)(Q(A + E)^T - Q(A)^T).$$

Consequently

$$||P_{A+E}-P_A|| \le ||Q(A+E)-Q(A)||\ ||Q(A+E)^T||+||Q(A)||\ ||Q(A+E)^T-Q(A)^T||.$$

Substitute $||Q(A)|| = ||Q(A + E)^T|| = 1$ and $||Q(A + E)^T - Q(A)^T|| = ||Q(A + E) - Q(A)||$ and obtain that $||P_{A+E} - P_A|| \le 2||Q(A + E) - Q(A)||$. Substitute the bound of Theorem 4.14. □

4.5.2 The basic algorithm

Assume an $m \times n$ matrix A having a small numerical rank r and a Gaussian $n \times r$ matrix H. Then according to [HMT11, Theorem 4.1], the column span of the matrices AH and $Q(AH)$ is likely to approximate the leading singular space $\mathbb{S}_{r,A}$ of the matrix A, and if it does, then it follows that the rank-r matrix QQ^TA approximates the matrix A.

In this subsection we recall the algorithm supporting this theorem, where temporarily we assume nonrandom multipliers H. In the next subsections we keep it nonrandom and estimate the output approximation errors of the algorithm assuming no oversampling, suggested in [HMT11]. Then we extend our study to the case where

H is a Gaussian, and in Section 4.5.5 cover the results in the case of random structured multipliers.

Algorithm 4.2. Low-rank approximation of a matrix. *(Cf. Remarks 4.5 and 4.6.)*

INPUT: *A matrix $A \in \mathbb{R}^{m\times n}$, its numerical rank r, and two integers $p \geq 2$ and $l = r + p \leq \min\{m, n\}$.*

OUTPUT: *an orthogonal matrix $Q \in \mathbb{R}^{m\times l}$ such that the matrix $QQ^TA \in \mathbb{R}^{m\times n}$ has rank at most l and approximates the matrix A.*

INITIALIZATION: *Generate an $n \times l$ matrix H.*

COMPUTATIONS:

1. *Compute an $n \times l$ orthogonal matrix $Q = Q(AH)$, sharing its range with the matrix AH.*
2. *Compute and output the matrix $R_{AH}A = QQ^TA$ and stop.*

This basic algorithm from [HMT11] uses $O(lmn)$ flops overall.

4.5.3 Analysis of the basic algorithm assuming no randomization and no oversampling

In Corollaries 4.9 and 4.10 of this subsection we estimate the error norms for the approximations computed by Algorithm 4.2 whose oversampling parameter p is set to 0, namely for the approximation of an orthogonal basis for the leading singular space $\mathbb{S}_{r,A}$ (by column set of the matrix Q of the algorithm) and for a rank-r approximation of the matrix A. We first recall the following results.

Theorem 4.15. *(Cf. (4.9).) Suppose A is an $m \times n$ matrix, $S_A \Sigma_A T_A^T$ is its SVD, r is an integer, $0 < r \le l \le \min\{m, n\}$, and $Q = Q_{r,A}$ is an orthogonal matrix basis for the space $\mathbb{S}_{r,A}$. Then $||A - QQ^T A|| = \sigma_{r+1}(A)$.*

Theorem 4.16. *Assume two matrices $A \in \mathbb{R}^{m \times n}$ and $H \in \mathbb{R}^{n \times r}$ and define the two matrices A_r and $\bar{A}_r$ of (4.7). Then $AH = A_r H + \bar{A}_r H$ where $A_r H = S_{r,A} U$, $U = \Sigma_{r,A} T_{r,A}^T H$. Furthermore the columns of the matrix $A_r H$ span the space $\mathbb{S}_{r,A}$ if* $\operatorname{rank}(A_r H) = r$.

These results together imply that the columns of the matrix $Q(AH)$ form an approximate orthogonal basis of the linear space $\mathbb{S}_A$, and next we estimate the error norms of this approximations.

Theorem 4.17. *Keep the assumptions of Theorem 4.16. Then*

(i) $||\bar{A}_r H||_F \le \sigma_{r+1}(A)\ ||H||_F$.

(ii) Furthermore if the matrix $T_{r,A}^T H$ is nonsingular, then

$$||(A_r H)^+|| \le ||(T_{r,A}^T H)^{-1}||/\sigma_r(A)$$

.

Proof. Recall that

$$||U|| = ||U||_F = 1,\ ||UAV|| \le ||A||, \text{ and } ||UAV||_F \le ||A||_F \tag{4.10}$$

for orthogonal matrices U and V Then note that $||\bar{A}_r H||_F = ||\bar{S}_{A,r} \bar{\Sigma}_{A,r} \bar{T}_{A,r}^T H||_F \le ||\bar{\Sigma}_{A,r} \bar{T}_{A,r}^T H||_F$ by virtue of bound (4.10).

Combine this bound with Lemma 4.2 and obtain that $||\bar{A}_r H||_F \le \sigma_{r+1}(A)\ ||\bar{T}_{A,r}^T H||_F$, which is not greater than $\sigma_{r+1}(A)\ ||H||_F$ by virtue of bound (4.10). This proves part (i).

Part (ii) follows because

$$(A_rH)^+ = (S_{r,A}\Sigma_{r,A}T_{r,A}^TH)^{-1} = (T_{r,A}^TH)^{-1}\Sigma_{r,A}^{-1}S_{r,A}^T$$

if the matrix $T_{r,A}^TH$ is nonsingular and because

$$||S_{r,A}|| = 1, and ||\Sigma_{r,A}^{-1}|| = 1/\sigma_r(A)$$

. □

Combine Theorems 4.14, 4.16, and 4.17 to obtain the following estimates.

Corollary 4.9. *Keep the assumptions of Theorem 4.16, let the matrix $T_{r,A}^TH$ be nonsingular and write*

$$||E||_F = \sigma_{r+1}(A)\ ||H||_F,$$

$$\Delta_+ = \sqrt{2}\ ||E||_F\ ||(T_{r,A}^TH)^{-1}||\ /\sigma_r(A),$$

and so

$$\Delta_+ = \sqrt{2}\ ||H||_F\ ||(T_{r,A}^TH)^{-1}||\ \sigma_{r+1}(A)/\sigma_r(A).$$

Then

$$\Delta = ||Q(A_rH)^T - Q(AH)^T|| \le \Delta_+ + O(||E||_F^2).$$

Next combine Corollary 4.8 with Theorem 4.15 and employ the orthogonal projection $P_{AH} = Q(AH)Q(AH)^T$ (cf. (4.9)) to extend the latter estimate to bound the error norm of low-rank approximation of a matrix A by means of Algorithm 4.2.

Corollary 4.10. *Keep the assumptions of Corollary 4.9 and write $\Delta'_+ = \sigma_{r+1}(A) + 2\Delta_+||A||$. Then*

$$\Delta' = ||A - P_{AH}A|| \le \Delta'_+ + O(||E||_F^2||A||).$$

Proof. Note that $||A-P_{AH}A|| \le ||A-P_MA||+||(P_M-P_{AH})A||$ for any $m\times r$ matrix M. Write $M = A_rH$, apply Theorem 4.15 and obtain $||A - P_MA|| = \sigma_{r+1}(A)$. Corollaries 4.8 and 4.9 together imply that $||(P_M - P_{AH})A|| \le ||A||\ ||P_{A_rH} - P_{AH}|| \le 2\Delta||A||$. Combine the above relationships. □

Remark 4.5. *Write $B_i = (A^TA)^iA$ and recall that $\sigma_j(B_i) = (\sigma_j(A))^{2i+1}$ for all positive integers i and j. Therefore one can apply the power transforms $A \to B_i$ for $i = 1, 2, \dots$ to increase the ratio $\sigma_r(A)/\sigma_{r+1}(A)$, which shows the gap between the two singular values. Consequently the bound Δ_+ on the error norm of the approximation of an orthogonal basis of the leading singular space $\mathbb{S}_{r,A}$ by $Q(B_iH)$ is expected to decrease as i increases (cf. [HMT11, equation (4.5)]). We use the matrix $AH = B_0H$ in Algorithm 4.2, but suppose we replace it with the matrices B_iH for small positive integer i, or even for $i = 1$, which would amount just to symmetrization. Then we would obtain low-rank approximation with the optimum error $\sigma_{r+1}(A)$ up to the terms of higher order in $\sigma_{r+1}(A)/\sigma_r(A)$ as long as the value $||H||_F||(T_{r,A}^TH)^{-1}||$ is reasonably bounded from above. The power transform $A = B_0 \to B_i$ requires to increase by a factor of $2i+1$ the number of matrix-by-vector multiplications involved, but for small positive integers i, the additional computational cost is still dominated by the costs of computing the SVD and rank-revealing factorizations.*

Remark 4.6. *Let us summarize our analysis. Suppose that the ratio $\sigma_r(A)/\sigma_{r+1}(A)$ is large and that the matrix product $P = T_{r,A}^TH$ has full rank r and is well-conditioned. Now set to 0 the oversampling integer parameter p of Algorithm 4.2. Then, by virtue of Theorem 4.17 and Corollaries 4.9 and 4.10, the algorithm outputs a close approximation $Q(AH)$ to an orthogonal bases for the leading singular space $\mathbb{S}_{r,A}$ of the input matrix A and a rank-r approximation to this matrix. Up to the terms of higher order, the error norm of the latter approximation is within a factor of $1 + ||H||_F\ ||(T_{r,A}^TH)^{-1}||/\sigma_r(A)$ from the optimal bound $\sigma_{r+1}(A)$. By applying the above*

power transform of the input matrix A at a low computational cost, we can decrease the error norm even below the value $\sigma_{r+1}(A)$.

4.5.4 Supporting low-rank approximation with Gaussian multipliers

In this subsection we extend the results of the previous one to support the choice of Gaussian multiplier H in Algorithm 4.2, whose "actual outcome is very close to the typical outcome because of the measure concentration effect" [HMT11, page 226].

Theorem 4.18. *Suppose $A \in \mathbb{R}^{m\times n}$, $A = S_A\Sigma_A T_A^T$ is its SVD of (4.1), $H = \mathbb{R}^{n\times r}$ is a Gaussian matrix, and* $\operatorname{rank}(A) = \rho \geq r$.

(i) Then the matrix $T_{r,A}^T H$ is Gaussian.

(ii) Assume the values Δ_+ and Δ'_+ of Corollaries 4.9 and 4.10 and the values $\nu_{F,n,r}$ and $\nu^+_{r,r}$ of Definition 4.2. Then $\Delta_+ = \sqrt{2}\ \nu_{F,n,r}\nu^+_{r,r}\sigma_{r+1}(A)/\sigma_r(A)$ and *$\Delta'_+ = \sigma_{r+1}(A) + 2\Delta_+||A||$.*

Proof. $T_A^T H$ is a Gaussian matrix by virtue of Lemma 4.3. Therefore so is its square submatrix $T_{r,A}^T H$ as well. This proves part (i), which implies part (ii). □

Corollary 4.11. *A Gaussian multiplier H is expected to support safe numerical application of Algorithm 4.2 even where the oversampling integer parameter p is set to 0.*

Proof. Combine Theorems 4.6 and 4.18 with Corollary 4.5. □

4.5.5 Supporting low-rank approximation with random structured multipliers

Multiplication of an $n \times n$ matrix A by a Gaussian matrix H at Stage 1 of Algorithm 4.2 requires $(2r-1)n^2$ flops, but we can save a factor of $r/\log(r)$ flops by applying structured random multipliers H. In particular we can use subsampled random Fourier

transforms (SRFTs) of [HMT11, equation (4.6)], subsampled random Hadamard transforms (SRHTs) of [T11], the chains of Givens rotations (CGRs) of [HMT11, Remark 4.5.1], and the leading Toeplitz submatrices $C_{n,r}$ and $C_{r,n}$ of random circulant $n \times n$ matrices C. We need just n random parameters to define a Gaussian circulant $n \times n$ matrix C and its leading Toeplitz blocks $C_{n,r}$ and $C_{r,n}$, and similarly for the other listed classes of structured matrices.

Example 4.3. *For two fixed integers l and n, $1 < l < n$, SRFT $n \times l$ matrices are the matrices of the form $S = \sqrt{n/l}\ D\Omega R$. Here D is a random $n \times n$ diagonal matrix whose diagonal entries are i.i.d. variables uniformly distributed on the unit circle $C(0,1) = \{x : |x| = 1\}$, Ω is the DFT matrix, and R is a random $n \times l$ permutation matrix defined by random choice of l columns under the uniform probability distribution on the set of the n columns of the identity matrix I_n (cf. [HMT11, equation (4.6) and Section 11]).*

Theorem 4.11 implies the following fact.

Corollary 4.12. *Assume an $n \times l$ SRFT matrix S. Then $\sqrt{l/n}\ \Omega^{-1}S$ is an $n \times l$ submatrix of a unitary circulant $n \times n$ matrix.*

According to the extensive tests by many researchers, various random structured $n \times l$ multipliers (such as SRFT, SRHT, CGR and CHR matrices) support low-rank approximation already where the oversampling parameter $p = l - r$ is a reasonable constant (see [HMT11] and [M11]). In particular SRFT with oversampling by 20 is adequate in almost all applications of low-rank approximations [HMT11, page 279]. Likewise, in our extensive tests covered in Section 4.6.2, Toeplitz multipliers defined as the $n \times r$ leading blocks of $n \times n$ random circulant matrices consistently supported low-rank approximation without oversampling as efficiently as Gaussian multipliers.

As in the case of our randomized support for GENP and block Gaussian elimination, formal analysis of the impact of random structured multipliers is complicated because we cannot use Lemma 4.3. Nevertheless, by allowing substantial oversampling, one can still prove that SRFT multipliers are expected to support low-rank approximation of a matrix having a small numerical rank.

Theorem 4.19. Error bounds for low-rank approximation with SRFT *(cf. [HMT11, Theorem 11.2]). Fix four integers* l, m, n, *and* r *such that* $4[\sqrt{r}+\sqrt{8\log(rn)n}]^2\log(r) \le l \le n$. *Assume an* $m \times n$ *matrix* A *with singular values* $\sigma_1 \ge \sigma_2 \ge \sigma_3 \ge \dots$, *an* $n \times l$ *SRFT matrix* S *of Example 4.3, and* $Y = AS$. *Then with a probability* $1 - O(1/r)$ *it holds that*

$$||(I - P_Y)A|| \le \sqrt{1 + 7n/l}\ \sigma_{r+1} \text{ and } ||(I - P_Y)A||_F \le \sqrt{1 + 7n/l}\ (\sum_{j>r} \sigma_j^2)^{1/2}.$$

Remark 4.7. *Clearly the theorem still holds if we replace the matrix* S *by the matrix* US *for a unitary matrix* $U = (1/\sqrt{n})\Omega^{-1}$. *In this case* $US = CR$ *for the matrix* R *of Example 4.3 and the circulant matrix* $C = \Omega^{-1}D\Omega$ *(cf. Theorem 4.11). By virtue of Theorem 4.19 we can expect that Algorithm 4.2 would produce a rank-*r *approximation if we choose a multiplier* H *being an SRFT* $n \times l$ *matrix or the* $n \times l$ *submatrix* CP *of* $n \times n$ *random unitary circulant matrix* C *made up of its* l *randomly selected columns where the selection is defined by the matrix* P *of Example 4.3 and where* l *is an integer of order* $r\log(r)$. *Recall that multiplication of an* $n \times n$ *Toeplitz matrix by an* $n \times l$ *matrix* $US = CP$ *involves* $O(nl\log(n))$ *flops [P01], versus* $O(n^2 l)$ *in the straightforward algorithm.*

4.6 Numerical Results

We performed numerical experiments with random general, circulant and Toeplitz matrices by using MATLAB in the Graduate Center of the City University of New York on a Dell computer with a Intel Core 2 2.50 GHz processor and 4G memory running Windows 7. In particular we generated Gaussian matrices by using the standard normal distribution function randn of MATLAB, and we use the MATLAB function rand for generating numbers in the range $[0, 1]$ under the uniform probability distribution function for Example 4.1. We display our estimates obtained in terms of the spectral matrix norm but our tests showed similar results where we used the Frobenius norm instead.

4.6.1 GENP with Gaussian and random circulant multipliers

We applied both GENP and the preprocessed GENP to $n \times n$ DFT matrices $A = \Omega$ and to the matrices A generate as follows. We fixed $n = 2^s$ and $k = n/2$ for $s = 6, 7, 8, 9, 10$, and first, by following [H02, Section 28.3], generated a $k \times k$ matrix $A_k = U\Sigma V^T$ where we chose $\Sigma = \text{diag}(\sigma_i)_{i=1}^k$ with $\sigma_i = 1$ for $i = 1, \dots, k-4$ and $\sigma_i = 0$ for $i = k-3, \dots, k$ and where U and V were $k \times k$ random orthonormal matrices, computed as the $k \times k$ factors $Q(X)$ in the QR factorization of $k \times k$ random matrices X. Then we generated Gaussian Toeplitz matrices B, C and D such that $||B|| \approx ||C|| \approx ||D|| \approx ||A_k|| \approx 1$ and defined the $n \times n$ matrix $A = \begin{pmatrix} A_k & B \\ C & D \end{pmatrix}$. For every dimension n, $n = 64, 128, 256, 512, 1024$ we run 1000 numerical tests where we solved the linear system $A\mathbf{x} = \mathbf{b}$ with Gaussian vector $\mathbf{b}$ and output the maximum, minimum and average relative residual norms $||A\mathbf{y} - \mathbf{b}||/||\mathbf{b}||$ as well as the standard deviation. Our Figure 1 and [PQYa, Table D.1] show the norms of A^{-1}. They ranged from 2.2×10^1 to 3.8×10^6 in our tests.

At first we describe the results of our tests for the latter class of matrices A. As we expected GEPP has always output accurate solutions to the linear systems $A\mathbf{y} = \mathbf{b}$ in our tests (see [PQYa, Table D.2]). GENP, however, was expected to fail for these systems, because the $(n/2) \times (n/2)$ leading principal block A_k of the matrix A was singular, having nullity $k - \text{rank}(A_k) = 4$. Indeed this caused poor performance of GENP in our tests, which have consistently output corrupted solutions, with relative residual norms ranging from 10^{-3} to 10^2.

In view of Corollary 4.7 we expected to fix this deficiency by means of multiplication by Gaussian matrices, and indeed in all our tests we observed residual norms below 1.3×10^{-6}, and they decreased below 3.6×10^{-12} in a single step of iterative refinement (see [PQYa, Table D.3]). Furthermore the tests showed the same power of preconditioning where we used the circulant multipliers of Examples 4.1 and 4.2 (see [PQYa, Tables D.4 and D.5]). As can be expected, the output accuracy of GENP with preprocessing has deteriorated a little versus GEPP in our tests. The output residual norms, however, were small enough to support application of the inexpensive iterative refinement. Already its single step decreased the average relative residual norm below 10^{-11} for $n = 1024$ in all our tests with Gaussian multipliers and to about 10^{-13} for $n = 1024$ in all our tests with circulant multipliers of Examples 4.1 and 4.2. See further details in our Figures 2 and 3 and [PQYa, Tables D.3–D.5]. This indicates that GENP with preprocessing followed by even a single step of iterative refinement is backward stable, similarly to the celebrated result of [S80].

We also applied similar tests to the $n \times n$ DFT matrix $A = \Omega$. The results were in very good accordance with our study in Section 4.4. Of course in this case the solution of a linear system $A\mathbf{x} = \mathbf{b}$ can be computed immediately as $\mathbf{x} = \frac{1}{n}\Omega^H\mathbf{b}$, but we were not seeking the solution, but were trying to compare the performance of GENP with and without preprocessing. In these tests the norm $||A^{-1}||$ was fixed at

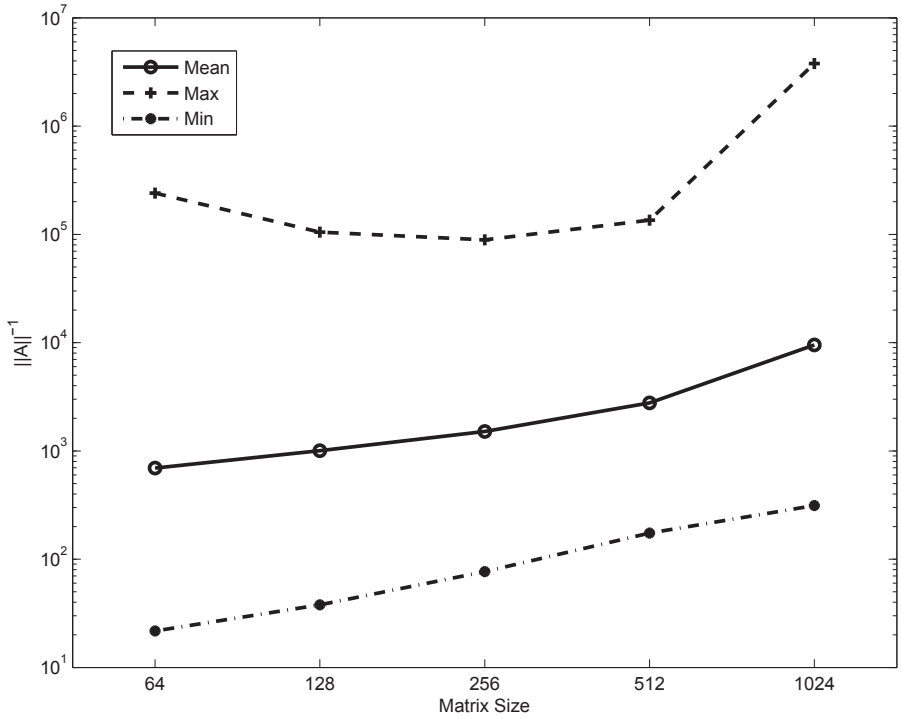

Figure 1: Norm of A^{-1}

$1/\sqrt{n}$. GEPP produced the solution within the relative residual norm between 10^{-15} and 10^{-16}, but GENP failed on the inputs Ω both when we used no preprocessing and used preprocessing with random circulant multipliers of Examples 4.1 and 4.2. In these cases the relative residual norms of the output approximations ranged between 10^{-2} and 10^4. In contrast GENP applied to the inputs preprocessed with Gaussian multipliers produced quite reasonable approximations to the solution. Already after a single step of iterative refinement, they have at least matched the level of GEPP. [PQYa, Table D.6]) displays these norms in some detail.

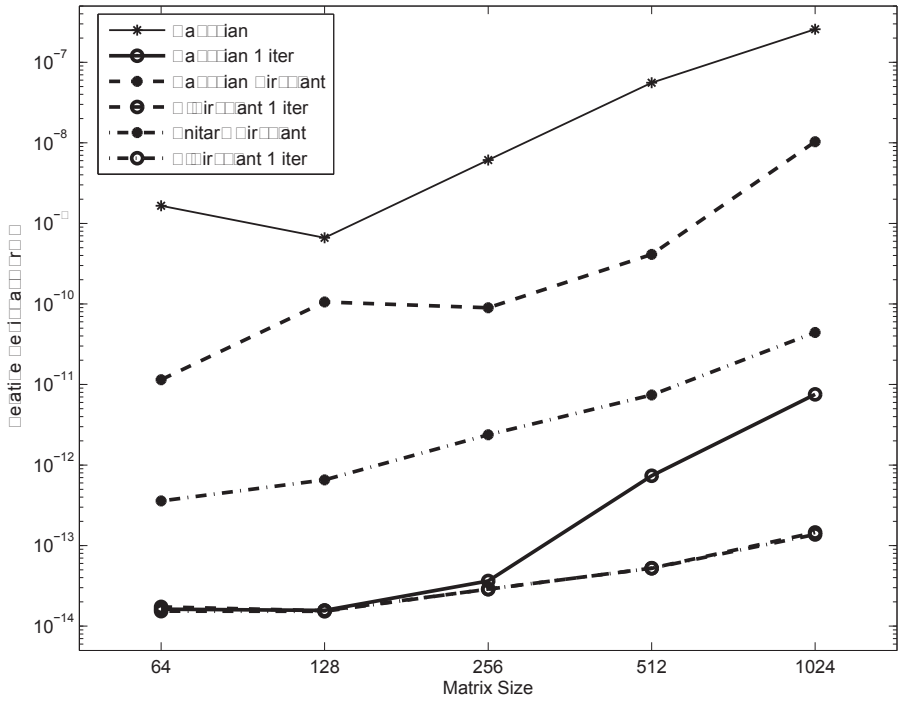

Figure 2: Average relative residual norms for GENP by using random multipliers. The two broken lines representing one iteration of circulant multipliers are overlapping at the bottom of the display

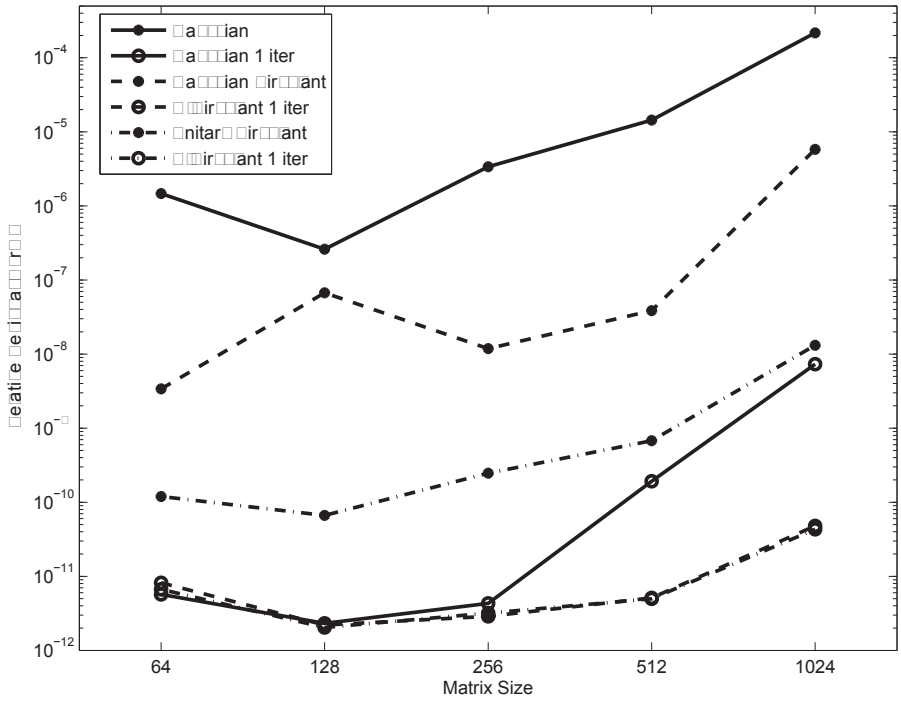

Figure 3: Maximum relative residual norms for GENP by using random multipliers. The two broken lines representing one iteration of circulant multipliers are overlapping at the bottom of the display

4.6.2 Approximation of the leading singular spaces and low-rank approximation of a matrix

We approximated the r-dimensional leading singular spaces of $n \times n$ matrices A that have numerical rank r, and we also approximated these matrices with matrices of rank r. For $n = 64, 128, 256, 512, 1024$ and $r = 8, 32$ we generated $n \times n$ random orthogonal matrices S and T and diagonal matrices $\Sigma = \mathrm{diag}(\sigma_j)_{j=1}^{n}$ such that $\sigma_j = 1/j,\ j = 1, \dots, r,\ \sigma_j = 10^{-10},\ j = r+1, \dots, n$ (cf. [H02, Section 28.3]). Then we computed the input matrices $A = S_A \Sigma_A T_A^T$, for which $||A|| = 1$ and $\kappa(A) = 10^{10}$. Furthermore we generated $n \times r$ random matrices H and computed the matrices $B_{r,A} = AH$, $Q_{r,A} = Q(B_{r,A})$, $S_{r,A}$, $T_{r,A}$, $Y_{r,A} = Q_{r,A}^T S_{r,A}$, and $Q_{r,A} Q_{r,A}^T A$. Our Figures 4–7 and [PQYa, Tables D.7–D.12] display the resulting data on the residual norms $\mathrm{rn}^{(1)} = ||Q_{r,A} Y_{r,A} - S_{r,A}||$ and $\mathrm{rn}^{(2)} = ||A - Q_{r,A} Q_{r,A}^T A||$, obtained in 1000 runs of our tests for every pair of n and r. In these figures and tables $\mathrm{rn}^{(1)}$ denotes the residual norms of the approximations of the matrix bases for the leading singular spaces $\mathbb{S}_{r,A}$, and $\mathrm{rn}^{(2)}$ denotes the residual norms of the approximations of the matrix A by the rank-r matrix $Q_{r,A} Q_{r,A}^T A$.

Our Figures 4 and 5 and [PQYa, Tables D.7–D.9] show the norm $\mathrm{rn}^{(1)}$. The last column of each of the tables displays the ratio of the observed values $\mathrm{rn}^{(1)}$ and its upper bound

$$\tilde{\Delta}_+ = \sqrt{2}\, \frac{\sigma_{r+1}(A)}{\sigma_r(A)} ||H||_F ||(T_{r,A}^T H)^{-1}||$$

estimated up to the higher order terms (cf. Corollary 4.9). In our tests we had $\sigma_r(A) = 1/r$ and $\sigma_{r+1}(A) = 10^{-10}$. [PQYa, Table D.7]) covers the case where we generated Gaussian multipliers H. [PQYa, Tables D.8 and D.9]) cover the cases where we generated random $n \times n$ circulant matrices of Examples 4.1 and 4.2 and applied their $n \times r$ Toeplitz leading blocks as multipliers H.

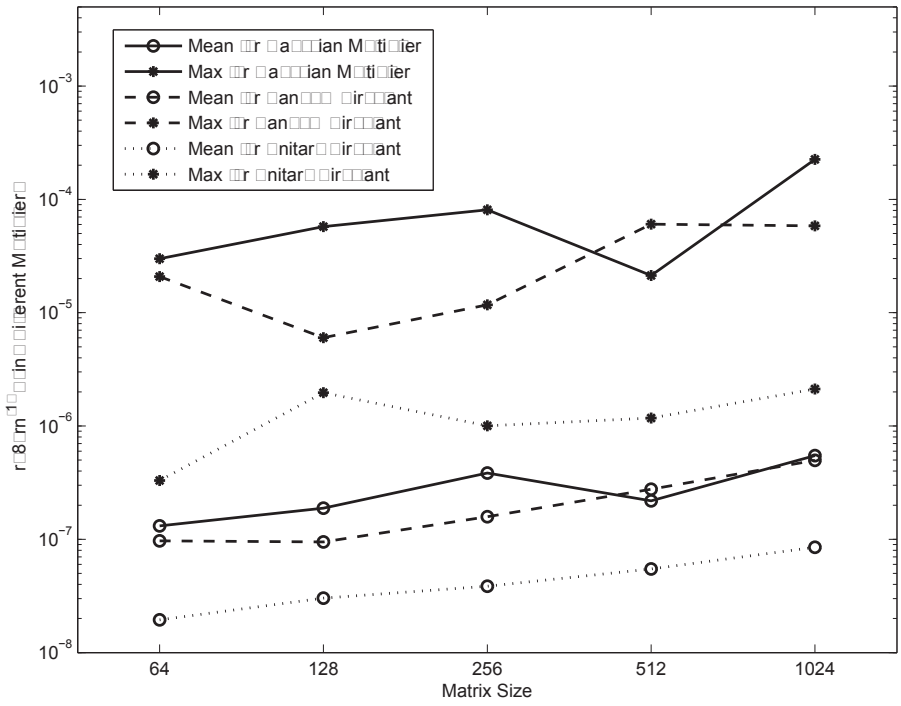

Figure 4: Residual norms $rn^{(1)}$ using different random multipliers, case r=8

Our Figures 6 and 7 and [PQYa, Table D.10–D.12] show similar results of our tests for the observed residual norms $\mathrm{rn}^{(2)}$ and their ratios with their upper bounds $\tilde{\Delta}'_+ = \sigma_{r+1}(A) + 2\Delta_+||A||$, estimated up to the higher order terms (cf. Corollary 4.10).

The test results are in quite good accordance with our theoretical study of Gaussian multipliers and suggest that the power of random circulant and Toeplitz multipliers is similar to the power of Gaussian multipliers, as in the case of various random structured multipliers of [HMT11] and [M11].

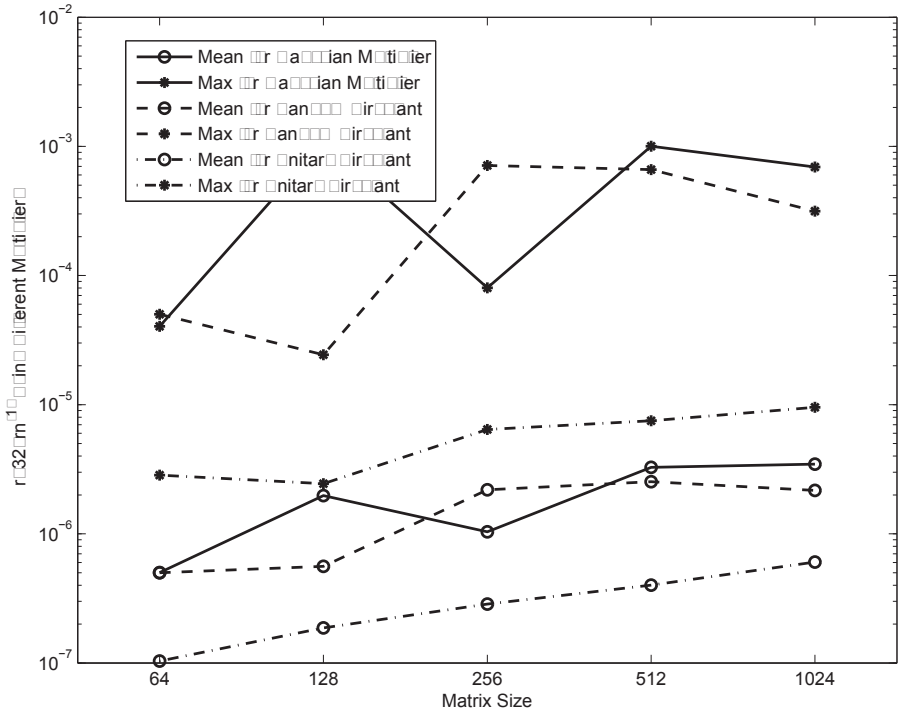

Figure 5: Residual norms $rn^{(1)}$ using different random multipliers, case r=32

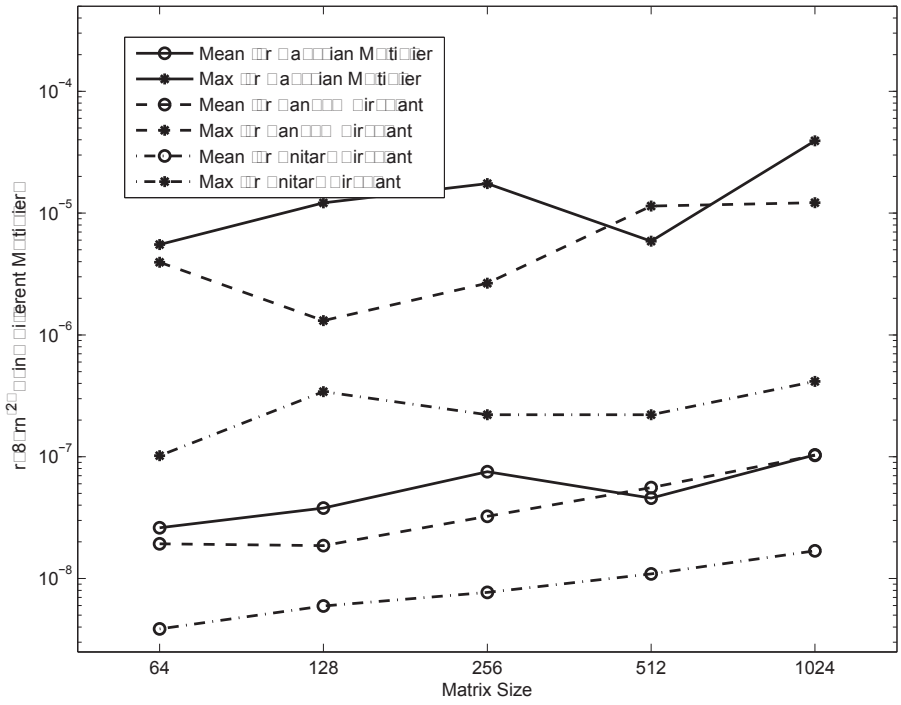

Figure 6: Residual norms $rn^{(2)}$ using different random multipliers, case r=8

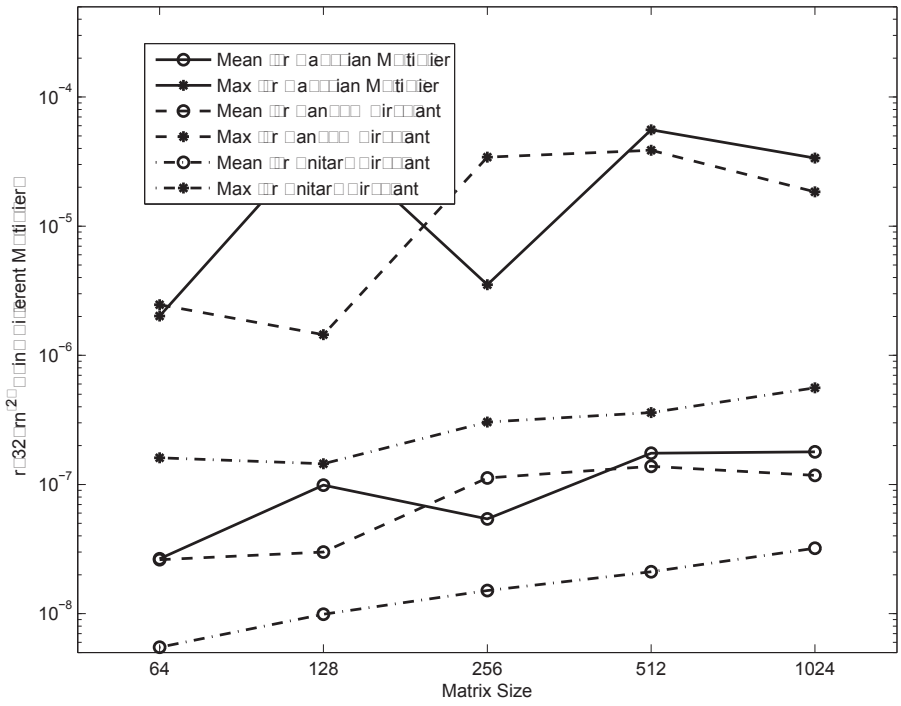

Figure 7: Residual norms $rn^{(2)}$ using different random multipliers, case r=32

References

[AR68] P. M. Anselone, L. B. Rall, The solution of the Characteristic Value-Vector Problems by Newton's Method, *Numerische Math.*, **11**, 38–45, 1968.

[A97] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1997.

[B75] S. Barnett, A Companion Matrix Analogue for Orthogonal Polynomials, *Linear Algebra and Appl.*, **12**, **3**, 197–208, 1975.

[B96] D. A. Bini, Numerical Computation of Polynomial Zeros by Means of Aberth's Method, *Numerical Algorithms*, **13**, 179–200, 1996.

[B02] M. Benzi, Preconditioning Techniques for Large Linear Systems: A Survey, *Journal of Computational Physics*, **182**, 418-477, 2002.

[B-S63] W. Börsch-Supan, A-posteriori Error Bounds for the Zeros of Polynomials, *Numer. Math.*, **5**, 380–398, 1963.

[BBCD93] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. Van Der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1993.

[BBD12] D. Becker, M. Baboulin, J. Dongarra, Reducing the amount of pivoting in symmetric indefinite systems, *Proceedings of the 9th International Conference on Parallel Processing and Applied Mathematics, PPAM 2011, Lecture Notes in Computer Science*, **7203**, 133–142, Springer-Verlag (2012). Also *INRIA* Research Report 7621 (05/2011), *University of Tennessee* Technical Report ICL-UT-11-06.

[BDDRvV00] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst, editors, *Templates for the solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia, 2000.

[BF00] D. A. Bini, G. Fiorentino, Design, Analysis and Implementation of a Multiprecision Polynomial Rootfinder, *Numerical Algorithms*, **23**, 127–173, 2000.

[BGTa] D. A. Bini, L. Gemignani, F. Tisseur, The Ehrlich-Aberth Method for the Nonsymmetric Tridiagonal Eigenvalue Problem, Numerical Analysis Report 428, *Dept. of Math., University of Pisa*, June 2003 (revised April 2004).

[BP88] D. Bini, V. Y. Pan, Efficient Algorithms for the Evaluation of the Eigenvalues of (Block) Banded Toeplitz Matrices, *Mathematics of Computation*, **50**, **182**, 431–448, 1988.

[BP91] D. Bini, V. Y. Pan, On the Evaluation of the Eigenvalues of a Banded Toeplitz Block Matrix, *J. of Complexity*, **7**, 408–424, 1991.

[CD05] Z. Chen, J. J. Dongarra, Condition Numbers of Gaussian Random Matrices, *SIAM. J. on Matrix Analysis and Applications*, **27**, 603–620, 2005.

[CN96] R. H. Chan, M. K. Ng, Conjugate Gradient Methods for Toeplitz Systems, *SIAM Review*, **38**, 427-482, 1996.

[CN99] R. H. Chan, M. K. Ng, Iterative Methods for Linear Systems with Matrix Structures, *Fast Reliable Algorithms for Matrices with Structure* (T. Kailath and A. H. Sayed, editors), 117–152, SIAM, Philadelpha, 1999.

[CGS07] S. Chandrasekaran, M. Gu, X. Sun, J. Xia, J. Zhu, A Superfast Algorithm for Toeplitz Systems of Linear Equations, *SIAM. J. on Matrix Analysis and Applications*, **29**, **4**, 1247–1266, 2007.

[CPW74] R. E. Cline, R. J. Plemmons, and G. Worm, Generalized Inverses of Certain Toeplitz Matrices, *Linear Algebra and Its Applications,* **8**, 25–33, 1974.

[D60] E. Durand, *Solutions numériques des équations algébriques*, *Tome 1: Equations du type F(X)=0; Racines d'un polynôme*, Masson, Paris, 1960.

[D88] J. Demmel, The Probability That a Numerical Analysis Problem Is Difficult, *Math. of Computation*, **50**, 449–480, 1988.

[DDSV98] I. S. Duff, A. M. Erisman, J. K. Reid, *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, UK, 1986.

[DER86] J. J. Dongarra, I. S. Duff, D. C. Sorensen, H. A. Van Der Vorst, *Numerical Linear Algebra for High-Performance Computers*, SIAM, Philadelphia, 1998.

[DL78] R. A. Demillo, R. J. Lipton, A Probabilistic Remark on Algebraic Program Testing, *Information Processing Letters*, **7**, **4**, 193–195, 1978.

[DS01] K. R. Davidson, S. J. Szarek, Local Operator Theory, Random Matrices, and Banach Spaces, in *Handbook on the Geometry of Banach Spaces* (W. B. Johnson and J. Lindenstrauss editors), pages 317–368, North Holland, Amsterdam, 2001.

[DZ07] J. Ding, A. Zhou, Eigenvalues of rank-one updated matrices with some applications, *Applied Mathematics Letters*, **20**,1223-1226, 2007.

[E88] A. Edelman, Eigenvalues and Condition Numbers of Random Matrices, *SIAM J. on Matrix Analysis and Applications*, **9**, **4**, 543–560, 1988.

[ES05] A. Edelman, B. D. Sutton, Tails of Condition Number Distributions, *SIAM J. on Matrix Analysis and Applications*, **27**, **2**, 547–560, 2005.

[G73] G. H. Golub, Some Modified Matrix Eigenvalue Problems, *SIAM Rev.*, **15**, 318–334, 1973.

[GE96] M. Gu, S. C. Eisenstat, Efficient Algorithms for Computing a Strong Rank-revealing QR Factorization, *SIAM Journal on Scientific Computing*, **17**, 848–869, 1996.

[GG03] J. von zur Gathen, J. Gerhard, *Modern Computer Algebra*, 2nd edition, Cambridge University Press, Cambridge, UK, 2003.

[GH90] J. R. Gilbert, H. Hafsteinsson, Parallel Symbolic factorization of Sparse Linear Systems, *Parallel Computing*, **14**, 151–162, 1990.

[GKO95] I. Gohberg, T. Kailath, V. Olshevsky, Fast Gaussian Elimination with Partial Pivoting for Matrices with Displacement Structure, *Mathematics of Computation*, **64**, 1557–1576, 1995.

[GL96] G. H. Golub, C. F. Van Loan, *Matrix Computations*, 3rd edition, The Johns Hopkins University Press, Baltimore, Maryland, 1996.

[GL13] G. H. Golub, C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Maryland, 2013 (fourth addition).

[GS92] J. R. Gilbert, R. Schreiber, Highly Parallel Sparse Cholesky Factorization, *SIAM J. Scientific Computing*, **13**, 1151–1172, 1992.

[GV00] G. H. Golub, H. A. van der Vorst, Eigenvalue Computation in the 20th Century, *Journal of Computational and Applied Mathematics*,**123**, 35-65, 2000.

[H64] A. Householder, *The Theory of Matrices in Numerical Analysis*, Dover, New York, 1964.

[H02] N. J. Higham, *Accuracy and Stability in Numerical Analysis*, SIAM, Philadelphia, 2002 (second edition).

[H04] Z. Huang, The Convergence Ball of Newton's Method and the Uniqueness Ball of Equations under Hölder-type Continuous Derivatives, *Computers and Math. (with Applications)*, **47**, 247–251, 2004.

[HP92] Y. P. Hong, C.–T. Pan, The Rank Revealing QR Decomposition and SVD, *Math. of Computation*, **58**, 213–232, 1992.

[HPR77] E. Hansen, M. Patrick, J. Rusnack, Some Modifications of Laguerre's Method, *BIT*, **17**, 409–417, 1977.

[HMT11] N. Halko, P. G. Martinsson, J. A. Tropp, Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions, *SIAM Review*, **53, 2**, 217–288, 2011.

[I97] I. C. F. Ipsen, Computing an Eigenvector with Inverse Iteration, *SIAM Review*, **39**, 354–391, 1998.

[JI92] . R. Jessup, I. C. F. Ipsen, Improving the accuracy of inverse iteration, *SIAM J. Sci. Stat. Comput.*, **13**,550–572, 1992.

[K66] I. O. Kerner, Ein Gesamtschrittverfahren zur Berechung der Nullstellen von Polynomen, *Numer. Math.*, **8**, 290–294, (1966).

[K70] V. N. Kublanovskaya, On an Approachto the solution ofthe Generalized Latent Value Problem for λ-Matrices, *SIAM J. on Numerical Analysis*, **7**, 542–537, 1970.

[K80] T. Kailath, *Linear Systems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

[KHG04] S.D. Kamvar, T.H. Haveliwala, G.H. Golub, Adaptive methods for the computation of pagerank, *Linear Algebra Appl.*, **386**, 51-65, 2004.

[LRT79] R. J. Lipton, D. Rose, R. E. Tarjan, Generalized Nested Dissection, *SIAM J. on Numerical Analysis*, **16**, **2**, 346–358, 1979.

[M73] K. Madsen, A Root-finding Algorithm Based on Newton's Method, *BIT*, **13**, 71–75, 1973.

[M97] A. Melman, A Unifying Convergence Analysis of Second-Order Methods for Secular Equations, *Math. Comp.*, **66**, 333–344, 1997.

[MR75] K. Madsen, J. Reid, Fortran Subroutines for Finding Polynomial Zeros, Report HL75/1172 (C.13), *Computer Science and Systems Division*, A. E. R. E. Harwell, Oxford, 1975.

[M00] R. Morgan, Preconditioning eigenvalues and some comparison of solvers, *Journal of Computational and Applied Mathematics*, **123**, 101-115, 2000.

[M11] M. W. Mahoney, Randomized Algorithms for Matrices and Data, *Foundations and Trends in Machine Learning*, NOW Publishers, **3, 2**, 2011. (Abridged version in: Advances in Machine Learning and Data Mining for Astronomy, edited by M. J. Way, et al., pp. 647-672, 2012.)

[NAG88] *NAG Fortran Library Manual*, Mark 13, Vol. **1**, 1988.

[O01] M. Overton, *Numerical Computing with IEEE Floating Point Arithmetic*, SIAM, Philadelphia, 2001.

[OR00] J. M. Ortega, W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, SIAM, Philadelphia, 2000.

[P00a] C.–T. Pan, On the Existence and Computation of Rank-revealing LU Factorization, *Linear Algebra and Its Applications*, **316**, 199–222, 2000.

[P01] V. Y. Pan, *Structured Matrices and Polynomials: Unified Superfast Algorithms*, Birkhäuser/Springer, Boston/New York, 2001.

[P07] V. Y. Pan, Null Aggregation and Extensions *Technical Report TR 200709, CUNY Ph.D. program in Computer Science*, April 2007.

[P15] V. Y. Pan, Transformations of Matrix Structures Work Again, *Linear Algebra and Its Applications*, 465, 1–32, 2015. Available at arxiv:1311.3729[math.NA]

[P80] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, New Jersey, 1980, and SIAM, Philadelphia, 1998.

[PB99] L Page, S Brin, R Motwani, T Winograd, *The pagerank citation ranking: Bringing order to the web*, http://dbpubs.stanford.edu , Stanford, CA, 1998

[PHI98] M. S. Petković, D. Herceg, S. Ilić, Safe Simultaneous Methods for Polynomial Zeros, *Numerical Algorithms*, **17**, 313–331, 1998.

[PIMRTY] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, Root-finding with eigen-solving, *Symbolic-Numeric Computation*, Birkhauser, Basel/Boston, 219–245, 2007.

[PIMRTTY] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, I. Taj-Eddin, X. Yan, Additive preconditioning and aggregation in matrix computations, *Computers & Mathematics with Applications*,**55**,1870–1886, 2008.

[PIM10] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, Additive Preconditioning for Matrix Computations, *Linear Algebra and Its Applications*, **432**, 1070–1089, 2010.

[PKMRTCY] V. Y. Pan, M. Kunin, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, W. Cao, X. Yan, Linking the TPR1, DPR1 and Arrow-Head Matrix Structures, *Computers & Mathematics with Applications*, **52**, 1603–1608, 2006.

[PP95] D. S. Parker, B. Pierce, The Randomizing FFT: An Alternative to Pivoting in Gaussian Elimination, Tech. Report CSD 950037, *Computer Science Dept., Univ. California at Los Angeles*, 1995.

[PQ10] V. Y. Pan, G. Qian, Randomized Preprocessing of Homogeneous Linear Systems of Equations, *Linear Algebra and Its Applications*, **432**, 3272–3318, 2010.

[PQ12] V. Y. Pan, G. Qian, Solving Linear Systems of Equations with Randomization, Augmentation and Aggregation, *Linear Algebra and Its Applications*, **437**, 2851–1876, 2012.

[PQYa] V. Y. Pan, G. Qian, X. Yan, Random Multipliers Numerically Stabilize Gaussian and Block Gaussian Elimination: Proofs and an Extension to Low-rank Approximation, arXiv:1406.5802v2 [math.NA] 17 Dec 2014.

Also see: Tech. Report TR 2013016, *PhD Program in Comp. Sci.*, *Graduate Center, CUNY*, 2013,

available at http://www.cs.gc.cuny.edu/tr/ techreport.php?id=463

[PQZ11] V. Y. Pan, G. Qian, A. Zheng, Randomized Preconditioning of the MBA Algorithm, in *Proc. International Symp. on Symbolic and Algebraic Computation (ISSAC'2011)*, San Jose, California, June 2011 (edited by Anton Leykin), 281–288, ACM Press, New York, 2011.

[PQZ13] V. Y. Pan, G. Qian, A. Zheng, Randomized Preprocessing versus Pivoting, *Linear Algebra and Its Applications*, **438, 4**, 1883–1899, 2013.

[PQZa] V. Y. Pan, G. Qian, A. Zheng, Randomized Matrix Computations, Tech. Report TR 2012009, *PhD Program in Comp. Sci.*, *Graduate Center, CUNY*, 2012.

Available at http://www.cs.gc.cuny.edu/tr/ techreport.php?id=438 and

http://arxiv.org/abs/1210.7476

[PQZb] V. Y. Pan, G. Qian, L. Zhao, Randomized Augmentation and Additive Preprocessing, preprint, 2014.

[PQZC] V. Y. Pan, G. Qian, A. Zheng, Z. Chen, Matrix Computations and Polynomial Root-finding with Preprocessing, *Linear Algebra and Its Applications*, 2014, DOI: 10.1016/j.laa.2014.06.027.

[PSZa] V. Y. Pan, J. Svadlenka, L. Zhao, Estimating the Norms of Circulant and Toeplitz Random Matrices and Their Inverses, *Linear Algebra and Its Applications*, 2014 (in press), DOI: 10.1016/j.laa.2014.06.027. Also arxiv:1311.3730[math.NA]

[PZa] V. Y. Pan, L. Zhao, A Single Random Triangular Toeplitz Multiplier Ensures Strong Nonsingularity, preprint, 2014.

[PR93] V. Y. Pan, J. Rief, Fast and Efficient Parallel Solution of Sparse Linear Systems, *SIAM J. on Computing*, **22**, **6**, 1227–1250, 1993.

[PY07] V. Y. Pan, X. Yan, Null Space and eigenspace computations with additive preprocessing, *Proceedings of the Third International Workshop on Symbolic-Numeric Computation*, 152–160, 2007.

[PY09] V. Y. Pan, X. Yan, Additive preconditioning, eigenspaces, and the inverse iteration, *Linear Algebra and its Applications*, **430**, 186–203, 2009.

[PVWC04] V. Y. Pan, M. Van Barel, X. Wang, G. Codevico, Iterative Inversion of Structured Matrices, *Theoretical Computer Science*, **315**, **2**–**3**, 581-592, 2004.

[PW79] G. Peters, J. H. Wilkinson, Inverse Iteration, Ill-Conditioned Equations and Newton's Method, *SIAM Review*, **21**, 339–360, 1979.

[S80] J. T. Schwartz, Fast Probabilistic Algorithms for Verification of Polynomial Identities, *Journal of ACM*, **27**, **4**, 701–717, 1980.

[S95] J.-G. Sun, On Perturbation Bounds for QR Factorization, *Linear Algebra and Its Applications*, **215**, 95–111, 1995.

[S98] G. W. Stewart, *Matrix Algorithms, Vol II: Eigensystems*, SIAM, Philadelphia, 1998.

[SST06] A. Sankar, D. Spielman, S.-H. Teng, Smoothed Analysis of the Condition Numbers and Growth Factors of Matrices, *SIAM J. on Matrix Analysis*, **28**, **2**, 446–476, 2006.

[ST95] T. R. Scavo, J. B. Thoo, On the Geometry of Halley's Method, *Amer. Math. Monthly*, **102**, 417–426, 1995.

[T11] J. A. Tropp, Improved Analysis of the Subsampled Randomized Hadamard Transform, *Adv. Adapt. Data Anal.*, **3, 1**–**2**, Special Issue, "Sparse Representation of Data and Images," 115-126, 2011.

[T64] J. F. Traub, *Iterative Methods for the Solution of Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1963.

[TM01] F. Tisseur, K. Meerbergen, The Quadratic Eigenvalue Problem, *SIAM Review*, **43**, **2**, 235–286, 2001.

[TB97] L. N. Trefethen, D. Bau, *Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997

[W02] D. S. Watkins *Fundamentals of Matrix Computations*, Wiley-Interscience, New Yor, 2002.

[W03] K. Weierstrass, *Neuer Beweis des Fundamentalsatzes der Algebra*, Mathematische Werker, Tome III, Mayer und Mueller, Berlin, 251–269, 1903.

[W65] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

[W68] V. Whitley, Certification of Algorithm 196: Muller's Method for Finding Roots of Arbitrary Function, *Comm. ACM*, **11**, 12–14, 1968.

[WZ95] D. Wang, F. Zhao, The Theory of Smale's Point Estimation and Its Application, *J. of Comput. and Applied Math.*, **60**, 253–269, 1995.

[Z79] R. E. Zippel, Probabilistic Algorithms for Sparse Polynomials, *Proceedings of EUROSAM'79, Lecture Notes in Computer Science*, **72**, 216–226, Springer, Berlin, 1979.